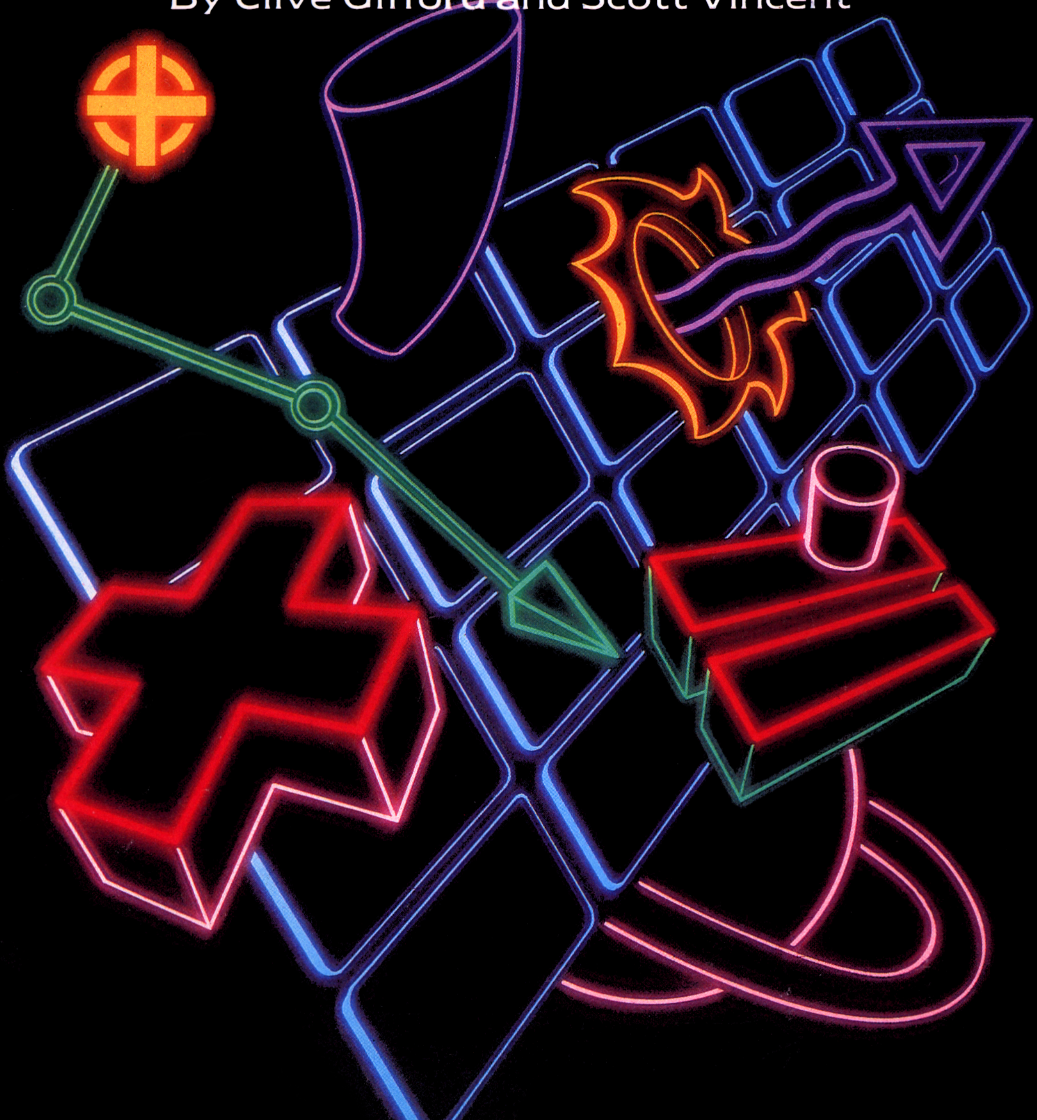
MACHINE CODE Routines for Your Amstrad

By Clive Gifford and Scott Vincent



MACHINE CODE ROUTINES FOR YOUR AMSTRAD

BY

SCOTT VINCENT

and

CLIVE GIFFORD

Best Wishes,

C. Gifford

Virgin

First published in Great Britain in 1985
by Virgin Books Ltd., 328 Kensal Rd,
London W10 5XJ

Copyright © 1985 by Scott Vincent and Clive Gifford

ISBN 0 86369 031 5

All rights reserved. No part of this book
may be reproduced in any form or by
any means without the prior permission of the publisher.

Designed by Ray Hyden.

Illustrations and layout by Sue Walliker and Ursula Shaw.

Photoset by Keyline Graphics.

Printed and bound in Great Britain by
Richard Clay (The Chaucer Press) Ltd,
Suffolk.

Distributed by Arrow Books.

Contents

Author's Introduction 5
Invert a Character 7
RSX Sound 10
Times 8 Print 17
Times 24 Print 20
Header Reader 24
Four Character Print 28
Screen Invert 31
Text Copy 35
Wrap Right 38
Wrap Left 42
Set Speed 45
Fast Screen Scroll (with Diamond Trail) 47 & 49
Box Scroller 51
Design Grid 53
Graphic Explosion 56
Screen Checker 63
Nine Character Print 65
Italics 69
Back Slant 72
Intelligent Copy 75
Screen Mover 77
Graphics Package 79
Useful PEEKs, POKEs, OUTs and CALLs. 93
Assemblers and Assembly Language 94
Bibliography 95
Keyboard Layout 96

ABOUT THE AUTHORS

Scott Vincent is a very experienced Machine Code programmer. Currently studying computer science at Sheffield University, Scott has had several Machine Code games published by well-known software houses. He has written several books which have been published by Interface and Virgin Books.

Clive Gifford has over 20 books in print and writes frequently for a number of computing magazines including *Home Computing Weekly*. His published titles include *The Amstrad Programmers Reference Guide* (co-authored with Tim Hartnell), *Using Computers in Education* and *Going On-Line With Your Computer*.

Together, the two authors have worked on a number of projects including several of the *Games for Your . . .* series of books published by Virgin.

The authors would like to thank the following people for their friendship: Dave, Dickie, Grannis, Lorraine, Simes and Stuart. Also, many thanks to Alan and Sue Baggs for their help with loan of equipment and John Brown for his agreement to the whole project.

AUTHORS' INTRODUCTION

The Amstrad range of computers are unequalled in the home computing world for their features to price ratio. The BASIC supplied with the Amstrad machines is an excellent one written by Locomotive Software, but no BASIC can ever match the speed of a Machine Code program. Machine Code has other advantages too. Well-written Machine Code programs can use far less memory space; moreover, routines can be written to perform tasks impossible through BASIC – witness the commercially available speech synthesisers and our faster cassette speed.

Machine Code does have its disadvantages, however. It is slow and tedious to write, has no error checking facilities (if there is an error in the program it will crash) and, of course, you need to be conversant with Machine Code to write any routines. To many people this is a task too great, and Machine Code remains a hallowed God, too difficult to learn.

This book is for those people unable to write Machine Code but capable of writing BASIC programs. We have tried to include routines of all types: ones covering graphics, sound, memory conservation, faster cassette retrieval and utilities. All have been supplied in the format of a BASIC loader, an Assembly listing, a detailed description of use and, in some cases, a demonstration program.

We like to think that this book will be an aid both to those BASIC programmers who have yet to delve into the intricacies of Machine Code and to some of those who have but are either having trouble or haven't come up with the same routines as we offer here.

Happy typing.

Clive Gifford
Scott Vincent

London, 1985

This book is dedicated to Catherine, Debbie, Louise and Sam.

Invert a Character

This routine turns your string of characters upside down. The program uses the first available User-Definable Graphic (UDG) character so two points must be borne in mind.

1) Make sure that there is at least one UDG character available; the number available can be altered by the SYMBOL AFTER command.

2) Make sure that you don't wish to use the first available UDG character; if you do want to use it then make SYMBOL AFTER, 1 lower.

The format to call this routine is as follows:

CALL 40000, X, Y, @A$

X and Y are the screen coordinates of where the first character is printed. As the string is to be inverted, the string will be printed from right to left across the screen. This should be remembered when specifying the start position coordinates. X and Y can be any numeric variable or any integer number providing it is within the screen range. Therefore CALL 40000, a, b, @A$ will work just as well as CALL 40000, 5, 9, @p$.

The string containing the characters can be any string variable but not a series of characters enclosed within speechmarks. Therefore, CALL 40000, X, Y, "HELLO" would not work but this is hardly a problem. A$ = "HELLO": CALL X, Y, @A$ would have the same effect.

The @ sign allows the routine to pass a string variable to and from Machine Code and is an extremely useful feature for the budding Machine Code programmer. This routine will work in all three of the Amstrad's screen modes.

```
1 '           INVERT A CHARACTER
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40076
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
```

```
50 DATA DD,6E,00,DD,66,01,7E,B7,C8,23
60 DATA 5E,23,56,DD,6E,02,DD,66,04,47
70 DATA C5,E5,CD,75,BB,1A,CD,64,9C,13
80 DATA E1,25,C1,10,F1,C9,D5,CD,A5,BB
90 DATA EB,CD,AE,BB,06,07,23,10,FD,F5
100 DATA CD,06,B9,0E,08,1A,06,08,1F,CB
110 DATA 16,10,FB,13,2B,0D,20,F3,CD,09
120 DATA B9,F1,CD,5A,BB,D1,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

                0010 ;              INVERT
                0020 ;
9C40            0030         ORG   40000
BB75            0040 CURSOR  DEFL  0BB75H
9C40 DD6E00     0050         LD    L,(IX+0)
9C43 DD6601     0060         LD    H,(IX+1)
9C46 7E         0070         LD    A,(HL)
9C47 B7         0080         OR    A
9C48 C8         0090         RET   Z
9C49 23         0100         INC   HL
9C4A 5E         0110         LD    E,(HL)
9C4B 23         0120         INC   HL
9C4C 56         0130         LD    D,(HL)
9C4D DD6E02     0140         LD    L,(IX+2)
9C50 DD6604     0150         LD    H,(IX+4)
9C53 47         0160         LD    B,A
9C54 C5         0170 NXTCHR  PUSH  BC
9C55 E5         0180         PUSH  HL
9C56 CD75BB     0190         CALL  CURSOR
9C59 1A         0200         LD    A,(DE)
9C5A CD649C     0210         CALL  INVCHR
9C5D 13         0220         INC   DE
```

```
9C5E E1       0230        POP  HL
9C5F 25       0240        DEC  H
9C60 C1       0250        POP  BC
9C61 10F1     0260        DJNZ NXTCHR
9C63 C9       0270        RET
9C64 D5       0280 INVCHR PUSH DE
BBA5          0290 MATRIX DEFL 0BBA5H
BBAE          0300 TABLE  DEFL 0BBAEH
B906          0310 ROMENA DEFL 0B906H
B909          0320 ROMDIS DEFL 0B909H
BB5A          0330 TXTOUT DEFL 0BB5AH
9C65 CDA5BB   0340        CALL MATRIX
9C68 EB       0350        EX   DE,HL
9C69 CDAEBB   0360        CALL TABLE
9C6C 0607     0370        LD   B,7
9C6E 23       0380 ADD7   INC  HL
9C6F 10FD     0390        DJNZ ADD7
9C71 F5       0400        PUSH AF
9C72 CD06B9   0410        CALL ROMENA
9C75 0E08     0420        LD   C,8
9C77 1A       0430 NXTBYT LD   A,(DE)
9C78 0608     0440        LD   B,8
9C7A 1F       0450 NXTBIT RRA
9C7B CB16     0460        RL   (HL)
9C7D 10FB     0470        DJNZ NXTBIT
9C7F 13       0480        INC  DE
9C80 2B       0490        DEC  HL
9C81 0D       0500        DEC  C
9C82 20F3     0510        JR   NZ,NXTBYT
9C84 CD09B9   0520        CALL ROMDIS
9C87 F1       0530        POP  AF
9C88 CD5ABB   0540        CALL TXTOUT
9C8B D1       0550        POP  DE
9C8C C9       0560        RET
              0570        END
```

RSX Sound

This amazing program gives you a similar set of commands to those found on the Oric computer. For those of you who are unfamiliar with the intricacies of the Oric (I imagine that is 99 per cent of you), one of its excellent sound features was its pre-defined sounds. If you typed in the BASIC command ZAP you got a ZAP sound; if you typed in EXPLODE you heard a most realistic explosion sound. This obviously saved programmers a great deal of time. Many published Oric listings make use of these handy sound commands.

Our routine creates a similar feature for the Amstrad. Six sounds (two more than the Oric) have been pre-defined, all of which we think will be of great use, particularly in games programs. After running the BASIC loader program, you must CALL 40000 to initialise the routine.

You will then have six new BASIC commands at your disposal. These have been created with the help of the Amstrad's ROM which allows new keywords to be created through a routine known as RSX. The six commands are:

¦EXPLODE
¦PING
¦SQUELCH
¦ZAP
¦ALIEN
¦SHOOT

The vertical bar preceding the names is part of the RSX feature and is the character found on the same key as '@' (to the right of 'P').

Happy zapping, squelching, and so on.

```
1 '                  RSX SOUND
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40315
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
```

```
50 DATA 01,4A,9C,21,5E,9C,CD,D1,BC,C9
60 DATA 62,9C,C3,82,9C,C3,A1,9C,C3,C0
70 DATA 9C,C3,DF,9C,C3,1B,9D,C3,4C,9D
80 DATA 00,00,00,00,45,58,50,4C,4F,44
90 DATA C5,53,51,55,45,4C,43,C8,50,49
100 DATA 4E,C7,5A,41,D0,41,4C,49,45,CE
110 DATA 53,48,4F,4F,D4,00,CD,A7,BC,3E
120 DATA 01,21,94,9C,CD,BC,BC,21,98,9C
130 DATA CD,AA,BC,C9,01,0F,FF,19,01,01
140 DATA 00,00,00,0F,0F,00,00,CD,A7,BC
150 DATA 06,0F,C5,78,32,BC,9C,21,B7,9C
160 DATA CD,AA,BC,30,FB,C1,10,F0,C9,01
170 DATA 01,00,00,00,00,0F,02,00,CD,A7
180 DATA BC,3E,01,21,D2,9C,CD,BC,BC,21
190 DATA D6,9C,CD,AA,BC,C9,01,0F,FF,05
200 DATA 01,01,00,19,00,00,0F,00,00,CD
210 DATA A7,BC,3E,01,21,05,9D,CD,BF,BC
220 DATA 21,09,9D,CD,AA,BC,06,0F,C5,3E
230 DATA 10,90,32,17,9D,21,12,9D,CD,AA
240 DATA BC,30,FB,C1,10,EE,C9,01,0F,04
250 DATA 02,01,00,01,28,00,00,06,1E,00
260 DATA 02,00,00,00,00,00,0F,02,00,CD
270 DATA A7,BC,3E,FF,21,33,9D,CD,BF,BC
280 DATA 21,3A,9D,CD,AA,BC,21,43,9D,CD
290 DATA AA,BC,C9,02,05,FF,02,05,01,02
300 DATA 01,00,01,64,00,00,07,64,00,02
310 DATA 00,01,6C,00,00,07,64,00,CD,A7
320 DATA BC,3E,01,21,66,9D,CD,BC,BC,21
330 DATA 6A,9D,CD,AA,BC,21,73,9D,CD,AA
340 DATA BC,30,FB,C9,01,0F,FF,02,01,00
350 DATA 00,00,00,0A,0F,04,00,01,01,00
360 DATA 00,00,05,0F,00,00
```

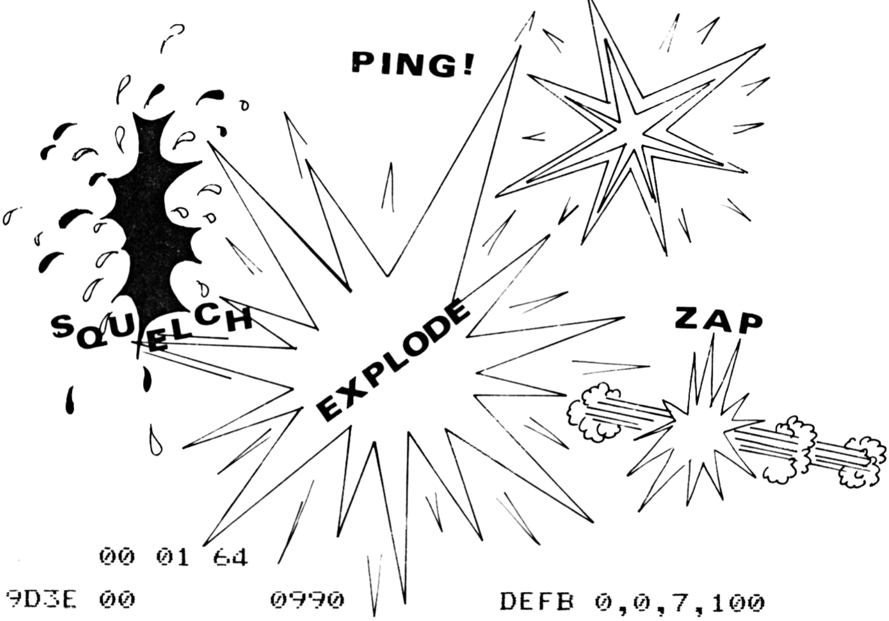
```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

                0001 ;      RSX  SOUND
                0002 ;
9C40            0010        ORG  40000
BCA7            0020 SNDRES DEFL 0BCA7H
BCBC            0030 AMPENV DEFL 0BCBCH
BCBF            0040 TONENV DEFL 0BCBFH
BCAA            0050 ADDSND DEFL 0BCAAH
BCD1            0060 LOGEXT DEFL 0BCD1H
9C40 014A9C     0070        LD   BC,TABLE
9C43 215E9C     0080        LD   HL,SPACE
9C46 CDD1BC     0090        CALL LOGEXT
9C49 C9         0100        RET
9C4A 629C       0110 TABLE  DEFW NAMETB
9C4C C3829C     0120        JP   EXPLOD
9C4F C3A19C     0130        JP   SQUELC
9C52 C3C09C     0140        JP   PING
9C55 C3DF9C     0150        JP   ZAP
9C58 C31B9D     0160        JP   ALIEN
9C5B C34C9D     0170        JP   SHOOT
9C5E 00         0180 SPACE  DEFB 0,0,0,0
     00 00 00
9C62            0190 NAMETB DEFM "EXPLOD"
9C68 C5         0200        DEFB "E"+80H
9C69            0210        DEFM "SQUELC"
9C6F C8         0220        DEFB "H"+80H
9C70            0230        DEFM "PIN"
9C73 C7         0240        DEFB "G"+80H
9C74            0250        DEFM "ZA"
9C76 D0         0260        DEFB "P"+80H
9C77            0270        DEFM "ALIE"
9C7B CE         0280        DEFB "N"+80H
9C7C            0290        DEFM "SHOO"
```

```
9C80 D4        0300        DEFB "T"+80H
9C81 00        0310        DEFB 0
9C82 CDA7BC    0320 EXPLOD CALL SNDRES
9C85 3E01      0330        LD   A,1
9C87 21949C    0340        LD   HL,AMP1
9C8A CDBCBC    0350        CALL AMPENV
9C8D 21989C    0360        LD   HL,SND1
9C90 CDAABC    0370        CALL ADDSND
9C93 C9        0380        RET
9C94 01        0390 AMP1   DEFB 1,15,255,25
     0F FF 19
9C98 01        0400 SND1   DEFB 1,1,0,0,0
     01 00 00 00
9C9D 0F        0410        DEFB 15,15,0,0
     0F 00 00
9CA1 CDA7BC    0420 SQUELC CALL SNDRES
9CA4 060F      0430        LD   B,15
9CA6 C5        0440 LOOP2  PUSH BC
9CA7 78        0450        LD   A,B
9CA8 32BC9C    0460        LD   (NOISE2),A
9CAB 21B79C    0470        LD   HL,SND2
9CAE CDAABC    0480 FULL2  CALL ADDSND
9CB1 30FB      0490        JR   NC,FULL2
9CB3 C1        0500        POP  BC
9CB4 10F0      0510        DJNZ LOOP2
9CB6 C9        0520        RET
9CB7 01        0530 SND2   DEFB 1,1,0,0,0
     01 00 00 00
9CBC 00        0540 NOISE2 DEFB 0,15,2,0
     0F 02 00
9CC0 CDA7BC    0550 PING   CALL SNDRES
9CC3 3E01      0560        LD   A,1
9CC5 21D29C    0570        LD   HL,AMP3
9CC8 CDBCBC    0580        CALL AMPENV
9CCB 21D69C    0590        LD   HL,SND3
9CCE CDAABC    0600        CALL ADDSND
```

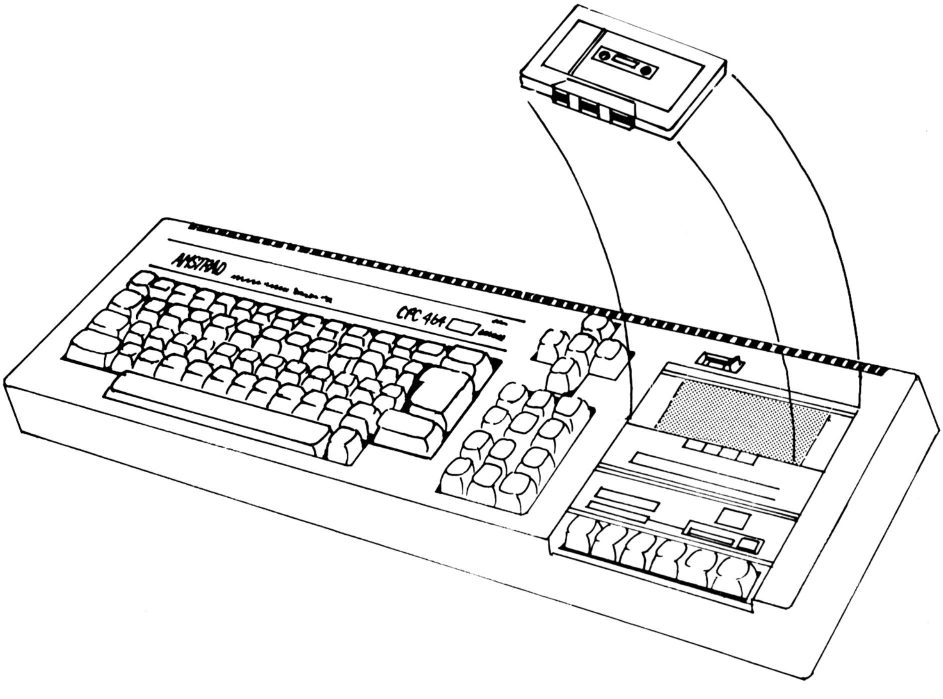
```
9CD1 C9         0610         RET
9CD2 01         0620 AMP3    DEFB 1,15,255,5
     0F FF 05
9CD6 01         0630 SND3    DEFB 1,1,0,25,0
     01 00 19 00
9CDB 00         0640         DEFB 0,15,0,0
     0F 00 00
9CDF CDA7BC     0650 ZAP     CALL SNDRES
9CE2 3E01       0660         LD   A,1
9CE4 21059D     0670         LD   HL,TONE4
9CE7 CDBFBC     0680         CALL TONENV
9CEA 21099D     0690         LD   HL,SND4A
9CED CDAABC     0700         CALL ADDSND
9CF0 060F       0710         LD   B,15
9CF2 C5         0720 LOOP4   PUSH BC
9CF3 3E10       0730         LD   A,16
9CF5 90         0740         SUB  B
9CF6 32179D     0750         LD   (NOISE4),A
9CF9 21129D     0760         LD   HL,SND4B
9CFC CDAABC     0770 FULL4   CALL ADDSND
9CFF 30FB       0780         JR   NC,FULL4
9D01 C1         0790         POP  BC
9D02 10EE       0800         DJNZ LOOP4
9D04 C9         0810         RET
9D05 01         0820 TONE4   DEFB 1,15,4,2
     0F 04 02
9D09 01         0830 SND4A   DEFB 1,0,1,40,0
     00 01 28 00
9D0E 00         0840         DEFB 0,6,30,0
     06 1E 00
9D12 02         0850 SND4B   DEFB 2,0,0,0,0
     00 00 00 00
9D17 00         0860 NOISE4 DEFB 0,15,2,0
     0F 02 00
9D1B CDA7BC     0870 ALIEN   CALL SNDRES
9D1E 3EFF       0880         LD   A,255
```

```
9D20 21339D    0890          LD   HL,TONE5
9D23 CDBFBC    0900          CALL TONENV
9D26 213A9D    0910          LD   HL,SND5A
9D29 CDAABC    0920          CALL ADDSND
9D2C 21439D    0930          LD   HL,SND5B
9D2F CDAABC    0940          CALL ADDSND
9D32 C9        0950          RET
9D33 02        0960 TONE5    DEFB 2,5,255,2
     05 FF 02
9D37 05        0970          DEFB 5,1,2
     01 02
9D3A 01        0980 SND5A    DEFB 1,0,1,100
```



```
     00 01 64
9D3E 00        0990          DEFB 0,0,7,100
     00 07 64
9D42 00        1000          DEFB 0
9D43 02        1010 SND5B    DEFB 2,0,1,108
     00 01 6C
9D47 00        1020          DEFB 0,0,7,100
     00 07 64
9D4B 00        1030          DEFB 0
9D4C CDA7BC    1040 SHOOT    CALL SNDRES
9D4F 3E01      1050          LD   A,1
9D51 21669D    1060          LD   HL,AMP6
```

```
9D54 CDBCBC    1070          CALL AMPENV
9D57 216A9D    1080          LD   HL,SND6A
9D5A CDAABC    1090          CALL ADDSND
9D5D 21739D    1100          LD   HL,SND6B
9D60 CDAABC    1110 FULL6    CALL ADDSND
9D63 30FB      1120          JR   NC,FULL6
9D65 C9        1130          RET
9D66 01        1140 AMP6     DEFB 1,15,255,2
     0F FF 02
9D6A 01        1150 SND6A    DEFB 1,0,0,0,0
     00 00 00 00
9D6F 0A        1160          DEFB 10,15,4,0
     0F 04 00
9D73 01        1170 SND6B    DEFB 1,1,0,0,0
     01 00 00 00
9D78 05        1180          DEFB 5,15,0,0
     0F 00 00
               1190          END
```



Times 8 Print

This routine transforms your tiny, insignificant 8 by 8 pixel character into a monster, 8 times as large – hence the imaginative title. The routine is, in fact, quite a simple one to write. What the routine does, in effect, is take your specified character, examine it line by line and wherever a pixel is set to the foreground colour, replace it with a complete solid block character, (CHR$(143)). This will make the resultant printed character 8 times larger than normal. Clever, eh?

The routine accepts the following command:

```
CALL 40000, X, Y, CHAR, P1, P2
```

X and Y are the top-left coordinates of where the enlarged character is to be printed from and CHAR is the ASCII code of the character to be enlarged. P1 and P2 control the colour of the top and bottom halves of the character respectively. It is thus possible to have the two halves different colours, creating quite a stunning effect.

The following demonstration routine will print the two characters, ‘GO’ in Mode 0 with different colour halves. To change the colours, simply press the spacebar.

```
10 MODE 0: CLS
20 FOR T = 0 TO 25
30 CALL 40000, 1, 8, 71, T, T+1
40 CALL 40000, 12, 8, 79, T+1, T
50 IF INKEY$ = “ ” THEN NEXT ELSE 50
60 END
```

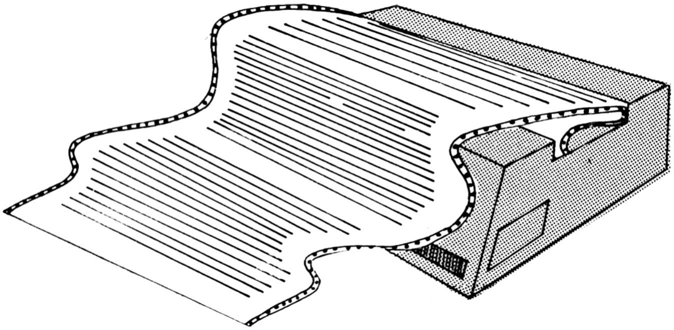
```
1 '              *8 PRINT
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40076
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
50 DATA CD,93,BB,F5,CD,06,B9,DD,7E,04
```

```
60 DATA CD,A5,BB,EB,DD,6E,06,DD,66,08
70 DATA 06,08,C5,E5,CD,75,BB,CB,40,28
80 DATA 05,DD,7E,00,18,03,DD,7E,02,CD
90 DATA 90,BB,E1,2C,0E,80,06,08,1A,A1
100 DATA 28,04,3E,8F,18,02,3E,20,CD,5A
110 DATA BB,CB,39,10,EF,13,C1,10,D1,CD
120 DATA 09,B9,F1,CD,90,BB,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

               0001 ;      *8   PRINT
               0002 ;
9C40           0010        ORG  40000
BB93           0020 GETPEN DEFL 0BB93H
BB90           0030 SETPEN DEFL 0BB90H
BBA5           0040 MATRIX DEFL 0BBA5H
BB75           0050 CURSOR DEFL 0BB75H
BB5A           0060 TXTOUT DEFL 0BB5AH
B906           0070 ROMENA DEFL 0B906H
B909           0080 ROMDIS DEFL 0B909H
9C40 CD93BB    0090        CALL GETPEN
9C43 F5        0100        PUSH AF
9C44 CD06B9    0110        CALL ROMENA
9C47 DD7E04    0120        LD   A,(IX+4)
9C4A CDA5BB    0130        CALL MATRIX
9C4D EB        0140        EX   DE,HL
9C4E DD6E06    0150        LD   L,(IX+6)
9C51 DD6608    0160        LD   H,(IX+8)
9C54 0608      0170        LD   B,8
9C56 C5        0180 NXTROW PUSH BC
9C57 E5        0190        PUSH HL
9C58 CD75BB    0200        CALL CURSOR
9C5B CB40      0210        BIT  0,B
```

```
9C5D 2805      0220        JR   Z,COL2
9C5F DD7E00    0230        LD   A,(IX+0)
9C62 1803      0240        JR   SETCOL
9C64 DD7E02    0250 COL2   LD   A,(IX+2)
9C67 CD90BB    0260 SETCOL CALL SETPEN
9C6A E1        0270        POP  HL
9C6B 2C        0280        INC  L
9C6C 0E80      0290        LD   C,128
9C6E 0608      0300        LD   B,8
9C70 1A        0310 NXTCOL LD   A,(DE)
9C71 A1        0320        AND  C
9C72 2804      0330        JR   Z,SPACE
9C74 3E8F      0340        LD   A,143
9C76 1802      0350        JR   PRINT
9C78 3E20      0360 SPACE  LD   A,32
9C7A CD5ABB    0370 PRINT  CALL TXTOUT
9C7D CB39      0380        SRL  C
9C7F 10EF      0390        DJNZ NXTCOL
9C81 13        0400        INC  DE
9C82 C1        0410        POP  BC
9C83 10D1      0420        DJNZ NXTROW
9C85 CD09B9    0430        CALL ROMDIS
9C88 F1        0440        POP  AF
9C89 CD90BB    0450        CALL SETPEN
9C8C C9        0460        RET
               0470        END
```



Times 24 Print

This is a variation on a theme. Instead of allotting one CHR$(143) character to each set pixel, this routine allots an extravagant 3 by 3 block of them, thus creating a character 3 times the size of the previous one and 24 times the size of a normal character. Obviously, the routine's effect will be limited if used in Mode 0 where there are only 20 columns across the screen.

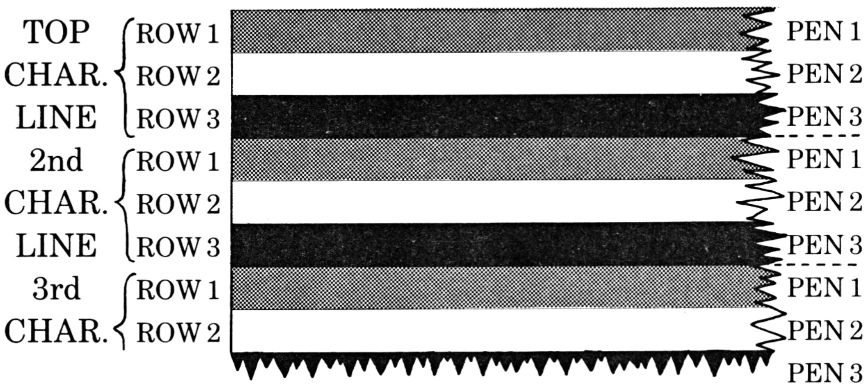
The routine does have one other difference as can be seen from the format below:

```
CALL 40000, X, Y, CHAR, P1, P2, P3
```

The extra pen parameter does make a major difference to the colour of the character. There were a number of ways that we could have utilised this extra colour but we decided to do it in the following way. Each line of the character is made up of 24 normal-sized characters across, by 3 rows down. The first row (24 characters across by 1 down) is always set to P1, the second row to P2 and the third to P3.

This is probably easier to show in a diagram, such as the one below:

PEN COLOURS ON TIMES 24 PRINT



AND SO ON

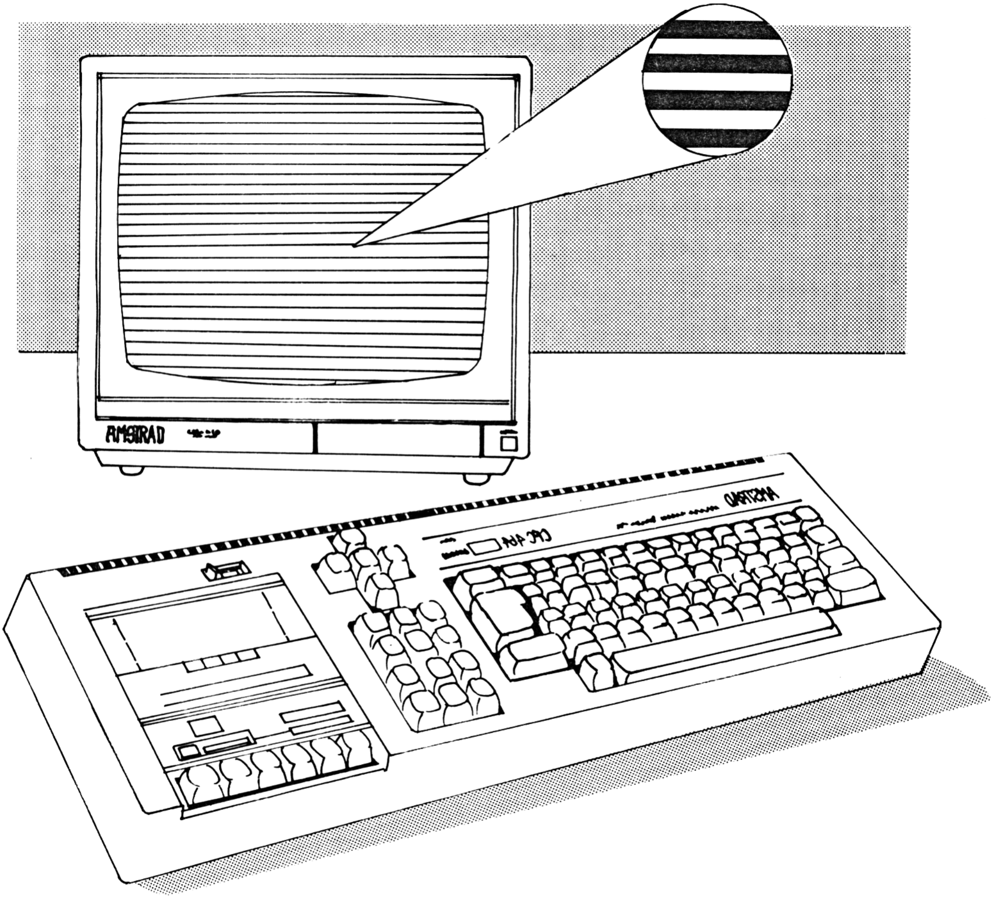
```
1 '              *24 PRINT
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40097
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
50 DATA CD,93,BB,F5,CD,06,B9,DD,7E,06
60 DATA CD,A5,BB,EB,DD,6E,08,DD,66,0A
70 DATA 06,08,C5,0E,80,06,03,C5,E5,CD
80 DATA 75,BB,CB,40,28,0E,CB,48,28,05
90 DATA DD,7E,04,18,08,DD,7E,00,18,03
100 DATA DD,7E,02,CD,90,BB,E1,06,08,1A
110 DATA A1,28,04,3E,8F,18,02,3E,20,CD
120 DATA 5A,BB,CD,5A,BB,CD,5A,BB,CB,39
130 DATA 10,E9,2C,C1,10,C5,13,C1,10,BC
140 DATA CD,09,B9,F1,CD,90,BB,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

                0001 ;         *24  PRINT
                0002 ;
9C40            0010           ORG  40000
BB93            0020 GETPEN    DEFL 0BB93H
BB90            0030 SETPEN    DEFL 0BB90H
BBA5            0040 MATRIX    DEFL 0BBA5H
BB75            0050 CURSOR    DEFL 0BB75H
BB5A            0060 TXTOUT    DEFL 0BB5AH
B906            0070 ROMENA    DEFL 0B906H
B909            0080 ROMDIS    DEFL 0B909H
9C40 CD93BB     0090           CALL GETPEN
9C43 F5         0100           PUSH AF
9C44 CD06B9     0110           CALL ROMENA
```

```
9C47 DD7E06    0120          LD   A,(IX+6)
9C4A CDA5BB    0130          CALL MATRIX
9C4D EB        0140          EX   DE,HL
9C4E DD6E08    0150          LD   L,(IX+8)
9C51 DD660A    0160          LD   H,(IX+10)
9C54 0608      0170          LD   B,8
9C56 C5        0180 NXTROW   PUSH BC
9C57 0E80      0190          LD   C,128
9C59 0603      0200          LD   B,3
9C5B C5        0210 AGAIN    PUSH BC
9C5C E5        0220          PUSH HL
9C5D CD75BB    0230          CALL CURSOR
9C60 CB40      0240          BIT  0,B
9C62 280E      0250          JR   Z,COL2
9C64 CB48      0260          BIT  1,B
9C66 2805      0270          JR   Z,COL3
9C68 DD7E04    0280          LD   A,(IX+4)
9C6B 1808      0290          JR   SETCOL
9C6D DD7E00    0300 COL3     LD   A,(IX+0)
9C70 1803      0310          JR   SETCOL
9C72 DD7E02    0320 COL2     LD   A,(IX+2)
9C75 CD90BB    0330 SETCOL   CALL SETPEN
9C78 E1        0340          POP  HL
9C79 0608      0350          LD   B,8
9C7B 1A        0360 NXTCOL   LD   A,(DE)
9C7C A1        0370          AND  C
9C7D 2804      0380          JR   Z,SPACE
9C7F 3E8F      0390          LD   A,143
9C81 1802      0400          JR   PRINT
9C83 3E20      0410 SPACE    LD   A,32
9C85 CD5ABB    0420 PRINT    CALL TXTOUT
9C88 CD5ABB    0430          CALL TXTOUT
9C8B CD5ABB    0440          CALL TXTOUT
9C8E CB39      0450          SRL  C
9C90 10E9      0460          DJNZ NXTCOL
9C92 2C        0470          INC  L
```

```
9C93 C1          0480           POP  BC
9C94 10C5        0490           DJNZ AGAIN
9C96 13          0500           INC  DE
9C97 C1          0510           POP  BC
9C98 10BC        0520           DJNZ NXTROW
9C9A CD09B9      0530           CALL ROMDIS
9C9D F1          0540           POP  AF
9C9E CD90BB      0550           CALL SETPEN
9CA1 C9          0560           RET
                 0570           END
```



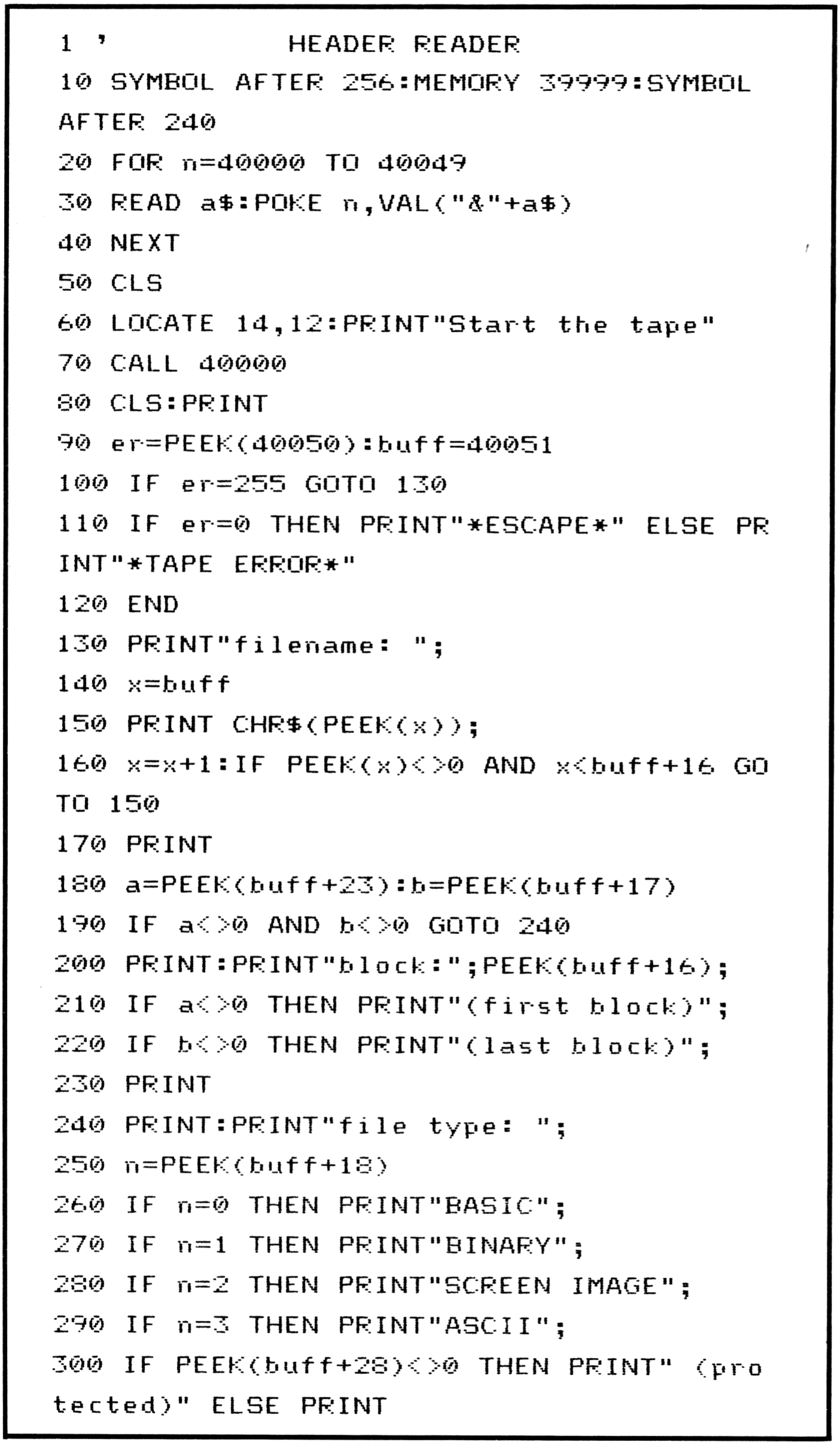
Header Reader

This short Machine Code routine in conjunction with a BASIC program enables you to obtain all the details you need about a program on cassette. You do not have to load the whole program in; the routine just requires the header of the program and from that it will find the following useful values:

File Name
Block Number (if file is more than one block long)
Specifies if it is the last block of a program
File Type (that is, BASIC, binary, ASCII file)
Specifies if the program is protected
Data Length
Data Location
Total File Length
Entry Address (only if it is binary)

As there is a BASIC program taking care of the Machine Code, there is no CALL routine. The program reads the header of the program into a 28 byte buffer specially created for this routine. The Machine Code then returns program control to the BASIC program, which looks at the buffer (with its PEEK command) and prints out the various pieces of information above.

You may find it difficult to think up an immediate use for such a program but they are extremely popular. The 'program breaker' packages that are advertised for getting into commercial software always feature a header reader among their suites of programs. Another use is for those programs of yours that you have saved on cassette but haven't documented (yes, it happens to all of us, particularly at two o'clock in the morning when you've just finished your mega-program). With this program you can obtain all the information that you need concerning your program without having to load the whole thing in again.



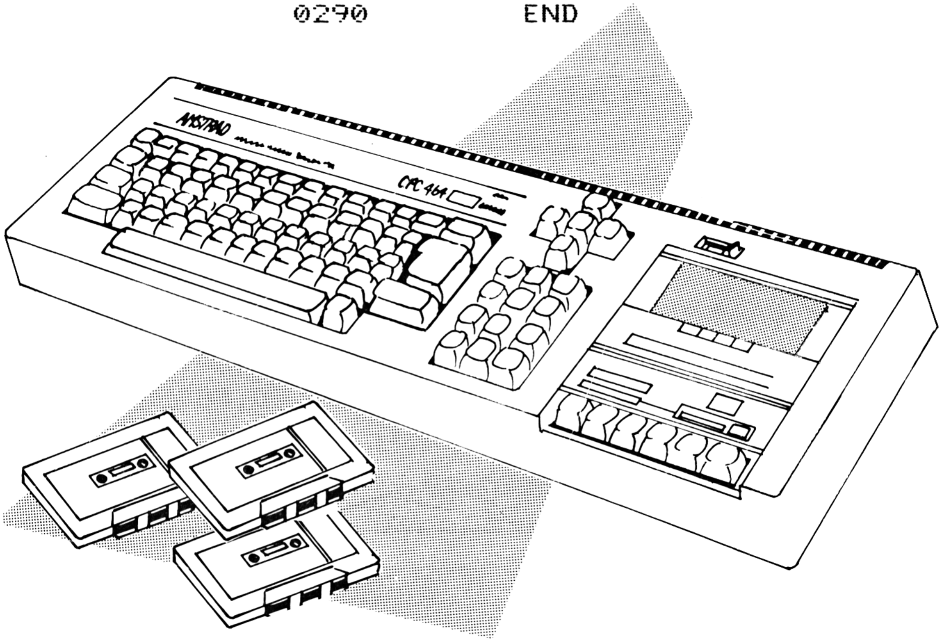
```
1 '               HEADER READER
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40049
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
50 CLS
60 LOCATE 14,12:PRINT"Start the tape"
70 CALL 40000
80 CLS:PRINT
90 er=PEEK(40050):buff=40051
100 IF er=255 GOTO 130
110 IF er=0 THEN PRINT"*ESCAPE*" ELSE PR
INT"*TAPE ERROR*"
120 END
130 PRINT"filename: ";
140 x=buff
150 PRINT CHR$(PEEK(x));
160 x=x+1:IF PEEK(x)<>0 AND x<buff+16 GO
TO 150
170 PRINT
180 a=PEEK(buff+23):b=PEEK(buff+17)
190 IF a<>0 AND b<>0 GOTO 240
200 PRINT:PRINT"block:";PEEK(buff+16);
210 IF a<>0 THEN PRINT"(first block)";
220 IF b<>0 THEN PRINT"(last block)";
230 PRINT
240 PRINT:PRINT"file type: ";
250 n=PEEK(buff+18)
260 IF n=0 THEN PRINT"BASIC";
270 IF n=1 THEN PRINT"BINARY";
280 IF n=2 THEN PRINT"SCREEN IMAGE";
290 IF n=3 THEN PRINT"ASCII";
300 IF PEEK(buff+28)<>0 THEN PRINT" (pro
tected)" ELSE PRINT
```

```
310 PRINT:PRINT"data location:";PEEK(buf
f+21)+256*PEEK(buff+22)
320 PRINT:PRINT"data length:";PEEK(buff+
19)+256*PEEK(buff+20)
330 IF n<>1 GOTO 370
340 PRINT:PRINT"entry address:";
350 nn=PEEK(buff+26)+256*PEEK(buff+27)
360 IF nn=0 THEN PRINT" NOT SET" ELSE PR
INT nn
370 PRINT:PRINT"total file length:";PEEK
(buff+24)+256*PEEK(buff+25)
380 LOCATE 3,18:PRINT"Press any key for
next header"
390 WHILE INKEY$="":WEND
400 GOTO 50
410 DATA 21,73,9C,11,1C,00,3E,2C,CD,A1
420 DATA BC,F5,CD,03,BB,F1,38,04,32,72
430 DATA 9C,C9,3E,FF,32,72,9C,3E,00,32
440 DATA 8F,9C,3A,85,9C,CB,47,28,03,32
450 DATA 8F,9C,E6,0E,CB,3F,32,85,9C,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

              0001 ;        HDR  READER
              0002 ;
9C40          0010          ORG  40000
BCA1          0020 CSREAD   DEFL 0BCA1H
BB03          0030 KMRES    DEFL 0BB03H
9C40 21739C   0040          LD   HL,BUFFER
9C43 111C00   0050          LD   DE,28
9C46 3E2C     0060          LD   A,2CH
9C48 CDA1BC   0070          CALL CSREAD
```

```
9C4B F5         0080         PUSH AF
9C4C CD03BB     0090         CALL KMRES
9C4F F1         0100         POP  AF
9C50 3804       0110         JR   C,OK
9C52 32729C     0120         LD   (ERROR),A
9C55 C9         0130         RET
9C56 3EFF       0140 OK      LD   A,255
9C58 32729C     0150         LD   (ERROR),A
9C5B 3E00       0160         LD   A,0
9C5D 328F9C     0170         LD   (PRTECT),A
9C60 3A859C     0180         LD   A,(BUFFER+18)
9C63 CB47       0190         BIT  0,A
9C65 2803       0200         JR   Z,NOPROT
9C67 328F9C     0210         LD   (PRTECT),A
9C6A E60E       0220 NOPROT  AND  14
9C6C CB3F       0230         SRL  A
9C6E 32859C     0240         LD   (BUFFER+18),A
9C71 C9         0250         RET
9C72 00         0260 ERROR   DEFB 0
001C            0270 BUFFER  DEFS 28
9C8F 00         0280 PRTECT  DEFB 0
                0290         END
```



Four Character Print

If you want a block of characters printed in a square (particularly used for User-Defined Graphics) then this is the routine for you. Quickly and painlessly, this routine will display a block on the screen.

The format is:

CALL 40000, X, Y, C

X and Y are the coordinates of the block's position on the graphics screen. Thus X can be anywhere between 0 and 639 and Y anywhere between 0 and 399. Note that the block's size limits the upper bounds of these coordinates' ranges by almost 20. The X and Y coordinates correspond to the topmost, leftmost pixel of the whole block.

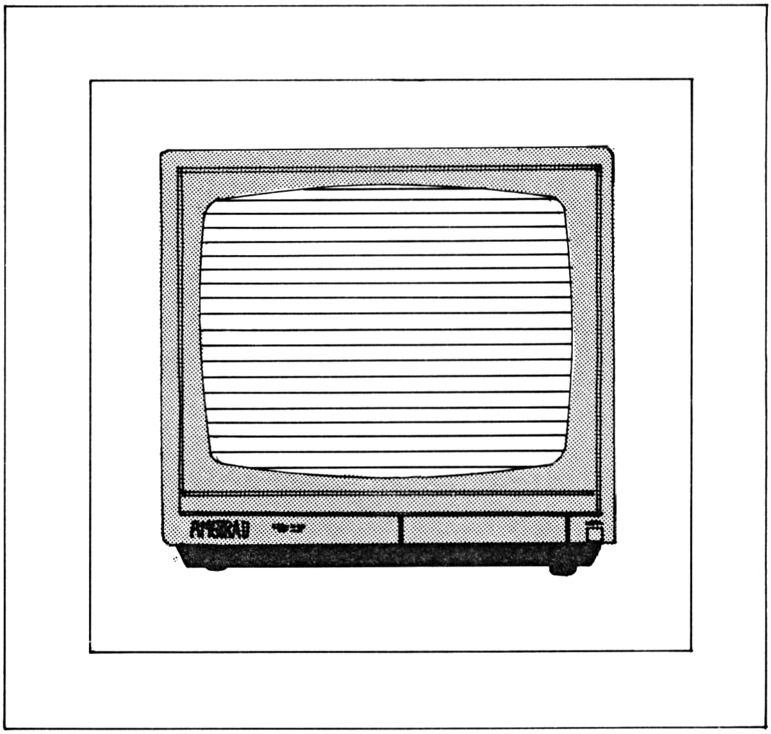
C corresponds to the ASCII code of the first character with the following 3 characters of the block being the next 3 characters in the ASCII range. This is extremely tidy and useful for user-defined graphics which will often use a different character for each part of a large block.

```
1 '            4 CHAR PRINT
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40048
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
50 DATA DD,4E,00,DD,6E,02,DD,66,03,DD
60 DATA 5E,04,DD,56,05,06,02,D5,E5,C5
70 DATA CD,C0,BB,C1,79,0C,C5,CD,FC,BB
80 DATA C1,79,0C,C5,CD,FC,BB,C1,E1,D1
90 DATA 3E,10,2B,3D,20,FC,10,E1,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

                   0001 ;         4CH   PRINT
                   0002 ;
9C40               0010           ORG   40000
BBC6               0020 ASKCUR DEFL 0BBC6H
BBC0               0030 MOVABS DEFL 0BBC0H
BBFC               0040 GRACHR DEFL 0BBFCH
9C40 DD4E00        0050           LD    C,(IX+0)
9C43 DD6E02        0060           LD    L,(IX+2)
9C46 DD6603        0070           LD    H,(IX+3)
9C49 DD5E04        0080           LD    E,(IX+4)
9C4C DD5605        0090           LD    D,(IX+5)
9C4F 0602          0100           LD    B,2
9C51 D5            0110 NXTCHR PUSH DE
9C52 E5            0120           PUSH HL
9C53 C5            0130           PUSH BC
9C54 CDC0BB        0140           CALL MOVABS
9C57 C1            0150           POP   BC
9C58 79            0160           LD    A,C
9C59 0C            0170           INC   C
9C5A C5            0180           PUSH BC
9C5B CDFCBB        0190           CALL GRACHR
9C5E C1            0200           POP   BC
9C5F 79            0210           LD    A,C
9C60 0C            0220           INC   C
9C61 C5            0230           PUSH BC
9C62 CDFCBB        0240           CALL GRACHR
9C65 C1            0250           POP   BC
9C66 E1            0260           POP   HL
9C67 D1            0270           POP   DE
9C68 3E10          0280           LD    A,16
9C6A 2B            0290 NEWROW DEC   HL
9C6B 3D            0300           DEC   A
```

```
9C6C 20FC      0310          JR   NZ,NEWROW
9C6E 10E1      0320          DJNZ NXTCHR
9C70 C9        0330          RET
               0340          END
```



Screen Invert

This routine simply rotates whatever is on the screen through 180 degrees, thus turning the whole screen upside down. It only works in Mode 1 but can be quite effective, as our demonstration program shows.

To access this routine, simply type:

CALL 40000

```
1 '                 SCREEN INVERT
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40143
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
50 DATA 21,00,00,CD,1A,BC,EB,2E,18,26
60 DATA 27,CD,1A,BC,23,7C,C6,38,67,0E
70 DATA 64,D5,E5,06,50,C5,1A,D5,56,06
80 DATA 08,5F,1F,7B,CB,12,1F,10,F8,06
90 DATA 04,0F,CB,0A,10,FB,72,D1,12,1C
100 DATA 20,0A,14,7A,E6,07,20,04,7A,D6
110 DATA 08,57,7D,2D,B7,20,0A,7C,25,E6
120 DATA 07,20,04,7C,C6,08,67,C1,10,C9
130 DATA E1,D1,7A,C6,08,57,E6,38,20,14
140 DATA 7A,D6,40,57,7B,C6,50,5F,30,0A
150 DATA 14,7A,E6,07,20,04,7A,D6,08,57
160 DATA 7C,D6,08,67,E6,38,FE,38,20,14
170 DATA 7C,C6,40,67,7D,D6,50,6F,30,0A
180 DATA 7C,25,E6,07,20,04,7C,C6,08,67
190 DATA 0D,20,86,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

                0001 ;       SCRN INVERT
                0002 ;
9C40            0010         ORG  40000
BC1A            0020 CHRPOS  DEFL 0BC1AH
9C40 210000     0030         LD   HL,0
9C43 CD1ABC     0040         CALL CHRPOS
9C46 EB         0050         EX   DE,HL
9C47 2E18       0060         LD   L,24
9C49 2627       0070         LD   H,39
9C4B CD1ABC     0080         CALL CHRPOS
9C4E 23         0090         INC  HL
9C4F 7C         0100         LD   A,H
9C50 C638       0110         ADD  A,56
9C52 67         0120         LD   H,A
9C53 0E64       0130         LD   C,100
9C55 D5         0140 NXTLIN  PUSH DE
9C56 E5         0150         PUSH HL
9C57 0650       0160         LD   B,80
9C59 C5         0170 NXTBYT  PUSH BC
9C5A 1A         0180         LD   A,(DE)
9C5B D5         0190         PUSH DE
9C5C 56         0200         LD   D,(HL)
9C5D 0608       0210         LD   B,8
9C5F 5F         0220 INVERT  LD   E,A
9C60 1F         0230         RRA
9C61 7B         0240         LD   A,E
9C62 CB12       0250         RL   D
9C64 1F         0260         RRA
9C65 10F8       0270         DJNZ INVERT
9C67 0604       0280         LD   B,4
9C69 0F         0290 ROTATE  RRCA
9C6A CB0A       0300         RRC  D
```

```
9C6C 10FB       0310        DJNZ ROTATE
9C6E 72         0320        LD   (HL),D
9C6F D1         0330        POP  DE
9C70 12         0340        LD   (DE),A
9C71 1C         0350        INC  E
9C72 200A       0360        JR   NZ,DEBTOK
9C74 14         0370        INC  D
9C75 7A         0380        LD   A,D
9C76 E607       0390        AND  7
9C78 2004       0400        JR   NZ,DEBTOK
9C7A 7A         0410        LD   A,D
9C7B D608       0420        SUB  8
9C7D 57         0430        LD   D,A
9C7E 7D         0440 DEBTOK LD   A,L
9C7F 2D         0450        DEC  L
9C80 B7         0460        OR   A
9C81 200A       0470        JR   NZ,HLBTOK
9C83 7C         0480        LD   A,H
9C84 25         0490        DEC  H
9C85 E607       0500        AND  7
9C87 2004       0510        JR   NZ,HLBTOK
9C89 7C         0520        LD   A,H
9C8A C608       0530        ADD  A,8
9C8C 67         0540        LD   H,A
9C8D C1         0550 HLBTOK POP  BC
9C8E 10C9       0560        DJNZ NXTBYT
9C90 E1         0570        POP  HL
9C91 D1         0580        POP  DE
9C92 7A         0590        LD   A,D
9C93 C608       0600        ADD  A,8
9C95 57         0610        LD   D,A
9C96 E638       0620        AND  56
9C98 2014       0630        JR   NZ,DELNOK
9C9A 7A         0640        LD   A,D
9C9B D640       0650        SUB  64
9C9D 57         0660        LD   D,A
```

```
9C9E 7B         0670        LD    A,E
9C9F C650       0680        ADD   A,80
9CA1 5F         0690        LD    E,A
9CA2 300A       0700        JR    NC,DELNOK
9CA4 14         0710        INC   D
9CA5 7A         0720        LD    A,D
9CA6 E607       0730        AND   7
9CA8 2004       0740        JR    NZ,DELNOK
9CAA 7A         0750        LD    A,D
9CAB D608       0760        SUB   8
9CAD 57         0770        LD    D,A
9CAE 7C         0780 DELNOK LD    A,H
9CAF D608       0790        SUB   8
9CB1 67         0800        LD    H,A
9CB2 E638       0810        AND   56
9CB4 FE38       0820        CP    56
9CB6 2014       0830        JR    NZ,HLLNOK
9CB8 7C         0840        LD    A,H
9CB9 C640       0850        ADD   A,64
9CBB 67         0860        LD    H,A
9CBC 7D         0870        LD    A,L
9CBD D650       0880        SUB   80
9CBF 6F         0890        LD    L,A
9CC0 300A       0900        JR    NC,HLLNOK
9CC2 7C         0910        LD    A,H
9CC3 25         0920        DEC   H
9CC4 E607       0930        AND   7
9CC6 2004       0940        JR    NZ,HLLNOK
9CC8 7C         0950        LD    A,H
9CC9 C608       0960        ADD   A,8
9CCB 67         0970        LD    H,A
9CCC 0D         0980 HLLNOK DEC   C
9CCD 2086       0990        JR    NZ,NXTLIN
9CCF C9         1000        RET
                1010        END
```

Text Copy

There is not a great deal to be said about this routine, except that the maxim 'small is beautiful' applies to it. This routine will dump the contents of a text screen to a printer connected up to your Amstrad.

This is useful in a number of situations such as outputting screenfuls of figures or data and is much easier than altering your program with hundreds of PRINT #8s to output each and every line of text to the printer instead of the screen. With this routine you still get the screen display and a fast hard copy of that screen.

To use this routine, just type CALL 40000 after you have RUN it.

```
1 '                TEXT COPY
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40054
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
50 DATA 2E,01,CD,17,BC,0C,04,C5,26,01
60 DATA E5,CD,75,BB,E1,CD,60,BB,CD,2E
70 DATA BD,38,FB,CD,31,BD,24,10,ED,2C
80 DATA 3E,0D,CD,2E,BD,38,FB,CD,31,BD
90 DATA 3E,0A,CD,2E,BD,38,FB,CD,31,BD
100 DATA C1,0D,20,D1,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

              0001 ;       TEXT COPY
              0002 ;
9C40          0010         ORG  40000
```

```
BC17            0020 CHRLIM DEFL 0BC17H
BB75            0030 CURSOR DEFL 0BB75H
BB60            0040 RDCHAR DEFL 0BB60H
BD2E            0050 BUSY   DEFL 0BD2EH
BD31            0060 SENDPR DEFL 0BD31H
9C40 2E01       0070        LD   L,1
9C42 CD17BC     0080        CALL CHRLIM
9C45 0C         0090        INC  C
9C46 04         0100        INC  B
9C47 C5         0110 NXTROW PUSH BC
9C48 2601       0120        LD   H,1
9C4A E5         0130 NXTCOL PUSH HL
9C4B CD75BB     0140        CALL CURSOR
9C4E E1         0150        POP  HL
9C4F CD60BB     0160        CALL RDCHAR
9C52 CD2EBD     0170 WAIT1  CALL BUSY
9C55 38FB       0180        JR   C,WAIT1
9C57 CD31BD     0190        CALL SENDPR
9C5A 24         0200        INC  H
9C5B 10ED       0210        DJNZ NXTCOL
9C5D 2C         0220        INC  L
9C5E 3E0D       0230        LD   A,13
9C60 CD2EBD     0240 WAIT2  CALL BUSY
9C63 38FB       0250        JR   C,WAIT2
9C65 CD31BD     0260        CALL SENDPR
9C68 3E0A       0270        LD   A,10
9C6A CD2EBD     0280 WAIT3  CALL BUSY
9C6D 38FB       0290        JR   C,WAIT3
9C6F CD31BD     0300        CALL SENDPR
9C72 C1         0310        POP  BC
9C73 0D         0320        DEC  C
9C74 20D1       0330        JR   NZ,NXTROW
9C76 C9         0340        RET
                0350        END
```

To complement this routine we have a demonstration which is probably one of the shortest word processors you will ever see. It simply accepts inputs of 80 characters on the mode 0 screen and leaves them on the screen until either a space is entered on its own as an input or until you reach the bottom of the screen. When either of these occur, the text copy routine is called and the screen is dumped to the printer. Put whatever printer control codes you like (such as double line spacing or emphasised printing) into line 1. This program may not be Wordstar, but then you can always improve on it yourself.

```
1 ' Add your own printer control codes
here
10 MODE 2:INK 1,26:INK 0,0:BORDER 13:PEN
 1:PAPER 0:CLS
20 FOR T=1 TO 24
30 LINE INPUT A$:IF A$=" " THEN CALL 400
00
40 NEXT
50 CALL 40000
```

Wrap Right

This textual scroll routine will scroll a row of text to the right with all pixels disappearing off the right-hand side of the screen reappearing on the left (thus creating a wrap-around effect).

Though it only works in Mode 1, the routine has a number of parameters to make it quite flexible in use.

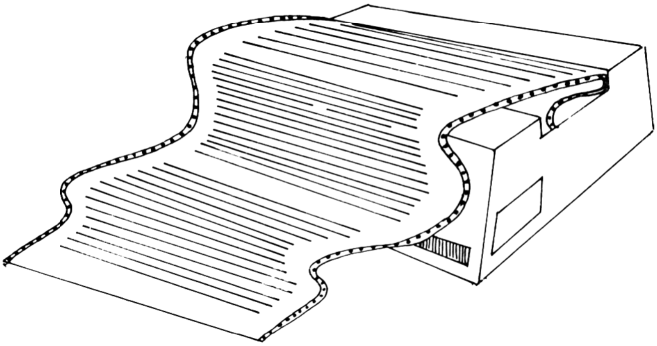
CALL 40000, X, Y, N

X equals the column and Y the row of where the scroll is going to start from. N equals the number of characters in that row that are to be scrolled to the right. Thus instead of indiscriminately scrolling large chunks of the screen, small pieces of essential text or, with UDGs and block graphics, graphic symbols for games can be scrolled precisely in their correct place.

You can have a number of areas scrolling on the screen simultaneously, although a CALL is needed to scroll each area. One CALL will only scroll the specified area by one pixel column so loops will have to be used to maintain a constant scrolling display.

Here is an example routine scrolling two areas of the Mode 1 screen. First, fill the screen up with characters of some description or another.

```
10 FOR T = 1 to 500
20 CALL 40000, 5, 4, 20
30 CALL 40000, 12, 10, 10
40 NEXT
```



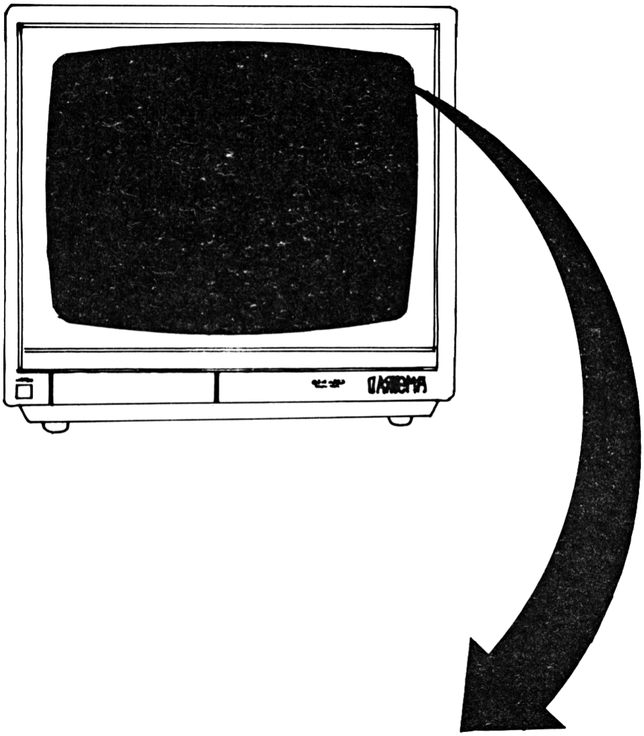
```
1 '                WRAP RIGHT
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40094
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
50 DATA FE,03,C0,DD,6E,02,2D,DD,4E,00
60 DATA DD,7E,04,81,D6,02,67,CD,1A,BC
70 DATA 23,41,CB,20,05,0E,08,E5,C5,CB
80 DATA 3E,7E,F5,5D,54,7D,2D,B7,20,0A
90 DATA 7C,25,E6,07,20,04,7C,C6,08,67
100 DATA 1A,CB,3E,38,04,CB,9F,18,02,CB
110 DATA DF,CB,5E,28,02,CB,FF,12,10,DB
120 DATA F1,38,04,CB,9E,18,02,CB,DE,CB
130 DATA 5F,28,02,CB,FE,C1,E1,7C,C6,08
140 DATA 67,0D,20,BD,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

                0001 ;       WRAP RIGHT
                0002 ;
9C40            0010         ORG  40000
BC1A            0020 CHRPOS  DEFL 0BC1AH
9C40 FE03       0030         CP   3
9C42 C0         0040         RET  NZ
9C43 DD6E02     0050         LD   L,(IX+2)
9C46 2D         0060         DEC  L
9C47 DD4E00     0070         LD   C,(IX+0)
9C4A DD7E04     0080         LD   A,(IX+4)
9C4D 81         0090         ADD  C
9C4E D602       0100         SUB  2
9C50 67         0110         LD   H,A
```

```
9C51 CD1ABC    0120         CALL CHRPOS
9C54 23        0130         INC  HL
9C55 41        0140         LD   B,C
9C56 CB20      0150         SLA  B
9C58 05        0160         DEC  B
9C59 0E08      0170         LD   C,8
9C5B E5        0180 NXTROW  PUSH HL
9C5C C5        0190         PUSH BC
9C5D CB3E      0200         SRL  (HL)
9C5F 7E        0210         LD   A,(HL)
9C60 F5        0220         PUSH AF
9C61 5D        0230 NXTBYT  LD   E,L
9C62 54        0240         LD   D,H
9C63 7D        0250         LD   A,L
9C64 2D        0260         DEC  L
9C65 B7        0270         OR   A
9C66 200A      0280         JR   NZ,HLOK
9C68 7C        0290         LD   A,H
9C69 25        0300         DEC  H
9C6A E607      0310         AND  7
9C6C 2004      0320         JR   NZ,HLOK
9C6E 7C        0330         LD   A,H
9C6F C608      0340         ADD  A,8
9C71 67        0350         LD   H,A
9C72 1A        0360 HLOK    LD   A,(DE)
9C73 CB3E      0370         SRL  (HL)
9C75 3804      0380         JR   C,SETBIT
9C77 CB9F      0390         RES  3,A
9C79 1802      0400         JR   TSTBIT
9C7B CBDF      0410 SETBIT  SET  3,A
9C7D CB5E      0420 TSTBIT  BIT  3,(HL)
9C7F 2802      0430         JR   Z,BITOK
9C81 CBFF      0440         SET  7,A
9C83 12        0450 BITOK   LD   (DE),A
9C84 10DB      0460         DJNZ NXTBYT
9C86 F1        0470         POP  AF
```

```
9C87 3804      0480          JR   C,WRAP
9C89 CB9E      0490          RES  3,(HL)
9C8B 1802      0500          JR   ENDPIX
9C8D CBDE      0510 WRAP     SET  3,(HL)
9C8F CB5F      0520 ENDPIX   BIT  3,A
9C91 2802      0530          JR   Z,ROWFIN
9C93 CBFE      0540          SET  7,(HL)
9C95 C1        0550 ROWFIN   POP  BC
9C96 E1        0560          POP  HL
9C97 7C        0570          LD   A,H
9C98 C608      0580          ADD  A,8
9C9A 67        0590          LD   H,A
9C9B 0D        0600          DEC  C
9C9C 20BD      0610          JR   NZ,NXTROW
9C9E C9        0620          RET
               0630          END
```



Wrap Left

Essentially the same as the last routine but with a scroll moving the opposite way.

The CALL statement is the same:

CALL 40000, X, Y, N

This routine works only in mode 1 and needs to be CALLed a number of times if it is going to scroll continuously. For more details, look at the Wrap Right routine.

```
1 '                 WRAP LEFT
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40087
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
50 DATA FE,03,C0,DD,6E,02,2D,DD,66,04
60 DATA 25,CD,1A,BC,DD,46,00,CB,20,05
70 DATA 0E,08,E5,C5,CB,26,7E,F5,5D,54
80 DATA 2C,20,0A,24,7C,E6,07,20,04,7C
90 DATA D6,08,67,1A,CB,26,38,04,CB,A7
100 DATA 18,02,CB,E7,CB,66,28,02,CB,C7
110 DATA 12,10,DD,F1,38,04,CB,A6,18,02
120 DATA CB,E6,CB,67,28,02,CB,C6,C1,E1
130 DATA 7C,C6,08,67,0D,20,BF,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

                0001 ;      WRAP LEFT
                0002 ;
9C40            0010        ORG  40000
BC1A            0020 CHRPOS DEFL 0BC1AH
9C40 FE03       0030        CP   3
9C42 C0         0040        RET  NZ
9C43 DD6E02     0050        LD   L,(IX+2)
9C46 2D         0060        DEC  L
9C47 DD6604     0070        LD   H,(IX+4)
9C4A 25         0080        DEC  H
9C4B CD1ABC     0090        CALL CHRPOS
9C4E DD4600     0100        LD   B,(IX+0)
9C51 CB20       0110        SLA  B
9C53 05         0120        DEC  B
9C54 0E08       0130        LD   C,8
9C56 E5         0140 NXTROW PUSH HL
9C57 C5         0150        PUSH BC
9C58 CB26       0160        SLA  (HL)
9C5A 7E         0170        LD   A,(HL)
9C5B F5         0180        PUSH AF
9C5C 5D         0190 NXTBYT LD   E,L
9C5D 54         0200        LD   D,H
9C5E 2C         0210        INC  L
9C5F 200A       0220        JR   NZ,HLOK
9C61 24         0230        INC  H
9C62 7C         0240        LD   A,H
9C63 E607       0250        AND  7
9C65 2004       0260        JR   NZ,HLOK
9C67 7C         0270        LD   A,H
9C68 D608       0280        SUB  8
9C6A 67         0290        LD   H,A
9C6B 1A         0300 HLOK   LD   A,(DE)
```

```
9C6C CB26      0310        SLA  (HL)
9C6E 3804      0320        JR   C,SETBIT
9C70 CBA7      0330        RES  4,A
9C72 1802      0340        JR   TSTBIT
9C74 CBE7      0350 SETBIT SET  4,A
9C76 CB66      0360 TSTBIT BIT  4,(HL)
9C78 2802      0370        JR   Z,BITOK
9C7A CBC7      0380        SET  0,A
9C7C 12        0390 BITOK  LD   (DE),A
9C7D 10DD      0400        DJNZ NXTBYT
9C7F F1        0410        POP  AF
9C80 3804      0420        JR   C,WRAP
9C82 CBA6      0430        RES  4,(HL)
9C84 1802      0440        JR   ENDPIX
9C86 CBE6      0450 WRAP   SET  4,(HL)
9C88 CB67      0460 ENDPIX BIT  4,A
9C8A 2802      0470        JR   Z,ROWFIN
9C8C CBC6      0480        SET  0,(HL)
9C8E C1        0490 ROWFIN POP  BC
9C8F E1        0500        POP  HL
9C90 7C        0510        LD   A,H
9C91 C608      0520        ADD  A,8
9C93 67        0530        LD   H,A
9C94 0D        0540        DEC  C
9C95 20BF      0550        JR   NZ,NXTROW
9C97 C9        0560        RET
               0570        END
```

Set Speed

This program could be titled 'Turbo Load and Save'; it can be used not only to increase the speed of saving and loading, but also to decrease it – perhaps for a security copy or to make a program appear longer than it really is.

Enough of these explanations and on with how the program works. The routine taps a routine within the Amstrad's ROM to alter the speed of loading and saving. With the Amstrad's BASIC, you can only select SPEED WRITE 0, (1000 baud) and SPEED WRITE 1, (2000 baud). With this routine you can select a far greater range of speeds. Above a certain speed, however, saving and loading successfully becomes increasingly unreliable. We found, by constant experimenting, that the fastest reliable speed was 3300 baud and it only offered good reliability when saving and loading on the same Amstrad machine.

For a complete explanation of the workings of the cassette system and terms such as half-zero length, refer to chapter 14.125 of the Firmware manual offered by Amsoft. All we will say here is that the format for this routine is:

CALL 40000, H, P

H is the Half Zero Length, but all we need to know here is that 333,333 (a third of a million) divided by this value will give you an approximate baud rate. Therefore, an H value of 167 would give you an approximate speed of 2000 baud.

P equals the Precompensation level (again, refer to the Firmware manual for a detailed explanation). Theoretically, this value can be between 0 and 255, though settings of between 2 and 100 seem more probable in practice. We've found a precompensation level of between 10 and 30 adequate.

If you don't specify any parameters after the CALL statement, then the computer will assign values of 100 to H and 10 to P. This gives an approximate speed of 3300 baud.

Please experiment with this routine, since different speeds may be suitable for your own particular machines.

```
1 '                    SET SPEED
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40026
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
50 DATA FE,00,20,07,21,64,00,3E,0A,18
60 DATA 0C,FE,02,C0,DD,7E,00,DD,6E,02
70 DATA DD,66,03,CD,68,BC,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

                0001 ;         SET   SPEED
                0002 ;
9C40            0010           ORG   40000
BC68            0020 SPEED     DEFL  0BC68H
9C40 FE00       0030           CP    0
9C42 2007       0040           JR    NZ,GTVALS
9C44 216400     0050           LD    HL,100
9C47 3E0A       0060           LD    A,10
9C49 180C       0070           JR    CONT
9C4B FE02       0080 GTVALS    CP    2
9C4D C0         0090           RET   NZ
9C4E DD7E00     0100           LD    A,(IX+0)
9C51 DD6E02     0110           LD    L,(IX+2)
9C54 DD6603     0120           LD    H,(IX+3)
9C57 CD68BC     0130 CONT      CALL  SPEED
9C5A C9         0140           RET
                0150           END
```

Fast Screen Scroll

For writing many types of arcade games and for exciting visual displays for almost any program, a fast, non-jerking scroll is a most useful addition to your collection of Machine Code routines.

This routine when called will scroll the whole screen either up or down by one character row. The format for the CALL is:

CALL 40000, C, UD

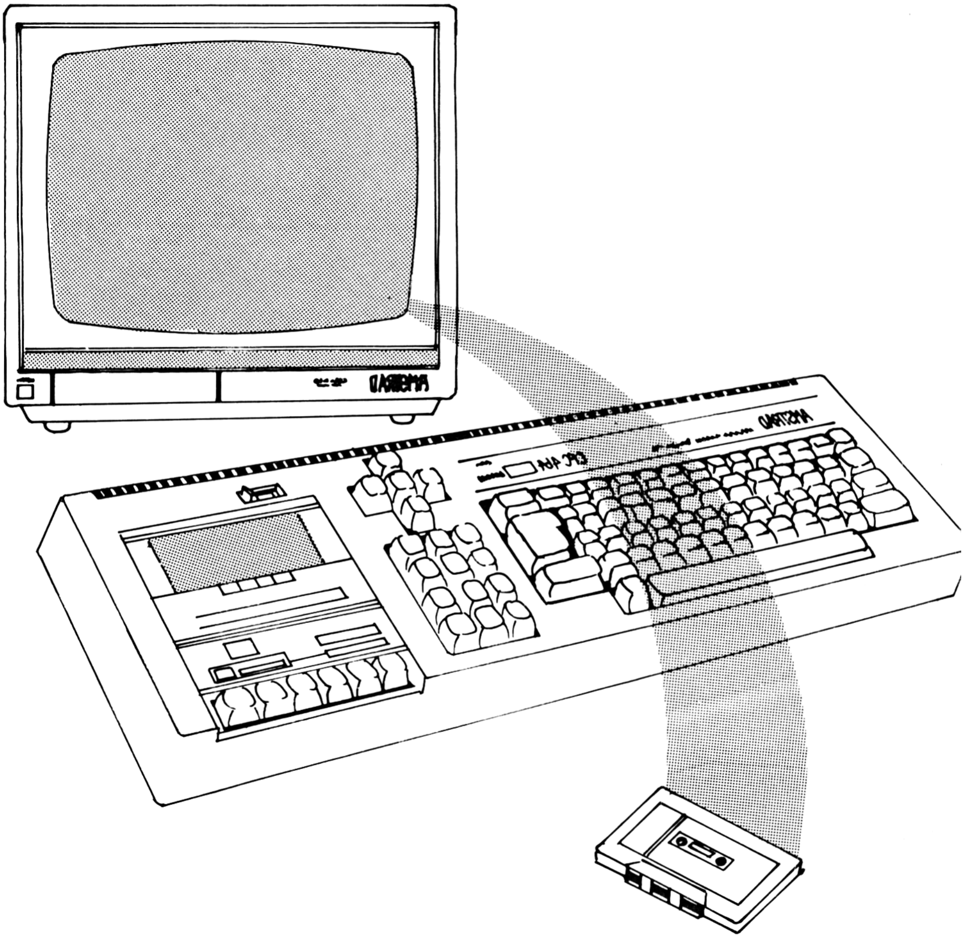
C is the ink colour of the new blank line which will appear at the top or the bottom of the screen depending on which scroll you use. The UD parameter defines the direction in which the scroll occurs; ‘0’ for down and ‘1’ for up.

```
1 '              SCREEN SCROLL
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40015
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
50 DATA FE,02,C0,DD,46,00,DD,7E,02,CD
60 DATA 2C,BC,CD,4D,BC,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

               0001 ;          SCR   SCROLL
               0002 ;
9C40           0010            ORG   40000
BC2C           0020 ENCODE DEFL  0BC2CH
BC4D           0030 HWROLL DEFL  0BC4DH
```

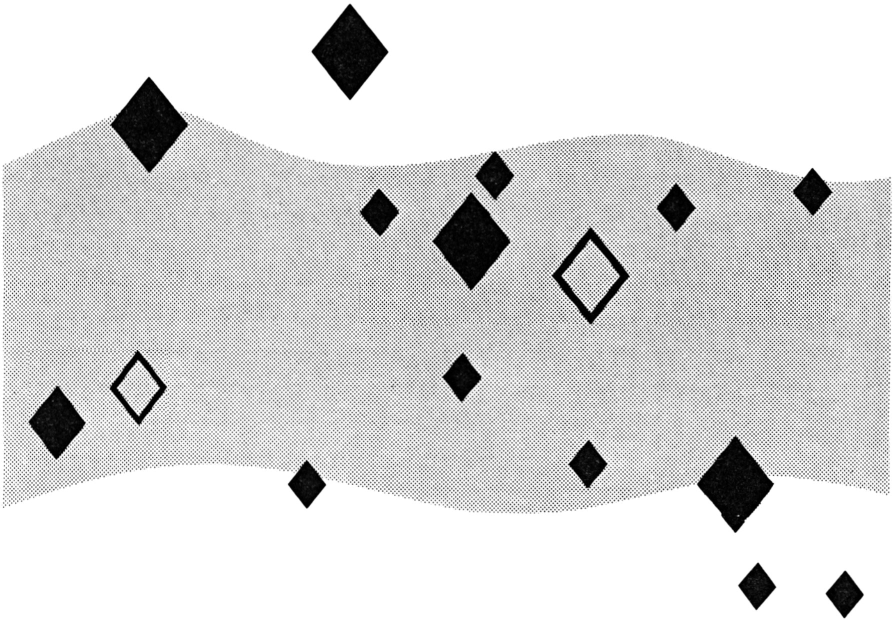
```
9C40 FE02       0040        CP   2
9C42 C0         0050        RET  NZ
9C43 DD4600     0060        LD   B,(IX+0)
9C46 DD7E02     0070        LD   A,(IX+2)
9C49 CD2CBC     0080        CALL ENCODE
9C4C CD4DBC     0090        CALL HWROLL
9C4F C9         0100        RET
                0110        END
```



Diamond Trail

To show the speed and the smoothness of this scroll, here is a short program that is a lot of fun to play. It is based on those scrolling road games where you must avoid going off the edge of the road. Our game is more involved than that. You are travelling along the famous Central African Diamond Trail looking for those elusive gems. The trail is a dangerous one and contains many potentially fatal hazards – indicated by a red cross. The glittering light blue diamonds are what you have come here for and by driving your car over them you pick them up and score an extra 50 points. You obtain 10 points for every tenth of a mile you travel and the game continues until you crash.

The game uses the fast scroll for moving the roadway towards you and it does this very neatly. The game also features an on-screen score and a high score. Go on . . . have a play.



```
10 RANDOMIZE TIME:hi=0
20 CLS:sc=0:x=3+INT(RND*24):y=10:c=x+y/2
30 x=3+INT(RND*24):c=x+y/2
40 LOCATE 1,1:PRINT STRING$(x,CHR$(143));STRING$(y," ");STRING$(40-x-y,CHR$(143))
50 IF RND<0.3 THEN PEN 3:LOCATE x+1+INT(RND*y),1:PRINT CHR$(203):PEN 1
60 IF RND<0.15 THEN PEN 2:LOCATE x+1+INT(RND*y),1:PRINT CHR$(227):PEN 1
70 LOCATE c,22:PRINT" "
80 c=c+(INKEY(8)=0)-(INKEY(1)=0)
90 CALL 40000,0,0
100 LOCATE 15,25:PRINT"SCORE =";sc
110 t=TEST(c*16-8,56)
120 IF t=1 OR t=3 GOTO 180
130 IF t=2 THEN SOUND 1,40,5,7:sc=sc+50
140 sc=sc+10:IF sc MOD 300=0 THEN y=y-1-(y=5)
150 LOCATE c,22:PRINT"^"
160 x=x+INT(RND*3)-1+(x>24)-(x<3)
170 GOTO 40
180 CLS:LOCATE 9,10:PRINT"CCCCRRRRAAAASSSSHHHH!!!!"
190 SOUND 1,1000,200,7
200 LOCATE 13,12:PRINT"You scored";sc
210 IF sc<hi GOTO 230
220 LOCATE 14,15:PRINT"NEW HIGH SCORE":hi=sc
230 LOCATE 13,20:PRINT"High Score =";hi
240 LOCATE 6,25:PRINT"Press SPACE BAR to play again"
250 WHILE INKEY$<>" ":WEND
260 GOTO 20
```

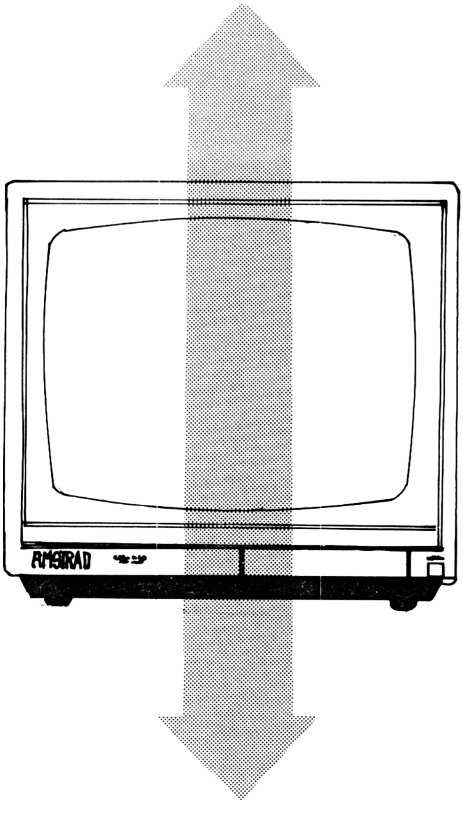
Box Scroller

This useful utility will scroll a portion of the screen up or down. The 'box' needs to be defined within the CALL statement, as does the ink colour and the direction in which the box will scroll; 1 for up and 0 for down.

When CALLed, the box will scroll by one complete character row. It performs the scroll extremely fast. The format for the CALL is:

CALL 40000, L, R, T, B, P, D

L is the left position, R is the right position, T is the top position and B is the bottom position of the box. This box is defined in a similar way to the BASIC WINDOW command. P is the ink colour and D stands for the direction of the scroll; '0' for down and '1' for up.



```
1 '              BOX SCROLL
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40031
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
50 DATA FE,06,C0,DD,46,00,DD,7E,02,DD
60 DATA 5E,04,DD,6E,06,DD,56,08,DD,66
70 DATA 0A,1D,2D,15,25,CD,2C,BC,CD,50
80 DATA BC,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

                0001 ;          BOX   SCROLL
                0002 ;
9C40            0010            ORG   40000
BC2C            0020 ENCODE     DEFL  0BC2CH
BC50            0030 SWROLL     DEFL  0BC50H
9C40 FE06       0040            CP    6
9C42 C0         0050            RET   NZ
9C43 DD4600     0060            LD    B,(IX+0)
9C46 DD7E02     0070            LD    A,(IX+2)
9C49 DD5E04     0080            LD    E,(IX+4)
9C4C DD6E06     0090            LD    L,(IX+6)
9C4F DD5608     0100            LD    D,(IX+8)
9C52 DD660A     0110            LD    H,(IX+10)
9C55 1D         0120            DEC   E
9C56 2D         0130            DEC   L
9C57 15         0140            DEC   D
9C58 25         0150            DEC   H
9C59 CD2CBC     0160            CALL  ENCODE
9C5C CD50BC     0170            CALL  SWROLL
9C5F C9         0180            RET
                0190            END
```

Design Grid

If you ever draw pictures on your Amstrad, or design screens of text and graphics for use in programs, then you'd find a block grid that overlays your screen at a touch of a button and then disappears leaving no blemish to your screen design at the touch of another button very useful.

Well this is exactly what Design Grid does using a special Machine Code feature called Exclusive OR'ing or XOR for short (strangely not from the planet Zilog).

CALLing 40000 will instantly display a grid of 8 by 8 pixels over the top of your screen design. This routine works in any screen mode and adjusts to that mode providing a grid containing many more lines in Mode 2 and many less in Mode 1.

CALL 40000 for the second time will remove the grid from the screen but will leave your design untouched – this is due to the Machine Code feature XOR.

The best way to use this routine is within an artist program where the routine is simply called by one button press. Following on from here is a short and simple BASIC artist program which under joystick control lets you draw a picture on the Mode 1 screen. Pressing the fire button will make the grid appear while pressing it for the second time will make it go again. I'm sure that you can write a far more complex drawing program or package; mine here is simply for demonstration purposes.

The grid was deliberately designed as an 8 by 8 pixel affair so as to correspond with a complete text character. It can therefore be used not only in graphical drawing programs but also in the detailed presentation of a text or semi-text screen.

```
1 '                 DESIGN GRID
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40083
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
50 DATA 3E,01,CD,59,BC,CD,11,BC,3C,0E
```

```
60 DATA 0A,CB,21,3D,20,FB,3E,01,CD,2C
70 DATA BC,11,07,00,41,C5,21,00,00,01
80 DATA C7,00,F5,D5,CD,62,BC,D1,F1,06
90 DATA 08,13,10,FD,C1,10,EA,69,26,00
100 DATA 29,29,29,2B,4D,44,21,00,00,1E
110 DATA 19,F5,D5,C5,E5,11,00,00,CD,5F
120 DATA BC,E1,06,08,23,10,FD,C1,D1,F1
130 DATA 1D,20,EA,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

                   0001 ;        DSGN GRID
                   0002 ;
9C40               0010          ORG  40000
BC59               0020 XORMOD DEFL 0BC59H
BC11               0030 MODE   DEFL 0BC11H
BC2C               0040 INKENC DEFL 0BC2CH
BC5F               0050 SCRHOR DEFL 0BC5FH
BC62               0060 SCRVER DEFL 0BC62H
9C40 3E01          0070          LD   A,1
9C42 CD59BC        0080          CALL XORMOD
9C45 CD11BC        0090          CALL MODE
9C48 3C            0100          INC  A
9C49 0E0A          0110          LD   C,10
9C4B CB21          0120 WIDTH    SLA  C
9C4D 3D            0130          DEC  A
9C4E 20FB          0140          JR   NZ,WIDTH
9C50 3E01          0150          LD   A,1
9C52 CD2CBC        0160          CALL INKENC
9C55 110700        0170          LD   DE,7
9C58 41            0180          LD   B,C
9C59 C5            0190 NXTVER PUSH BC
9C5A 210000        0200          LD   HL,0
9C5D 01C700        0210          LD   BC,199
```

```
9C60 F5       0220         PUSH AF
9C61 D5       0230         PUSH DE
9C62 CD62BC   0240         CALL SCRVER
9C65 D1       0250         POP  DE
9C66 F1       0260         POP  AF
9C67 0608     0270         LD   B,8
9C69 13       0280 MOVEX   INC  DE
9C6A 10FD     0290         DJNZ MOVEX
9C6C C1       0300         POP  BC
9C6D 10EA     0310         DJNZ NXTVER
9C6F 69       0320         LD   L,C
9C70 2600     0330         LD   H,0
9C72 29       0340         ADD  HL,HL
9C73 29       0350         ADD  HL,HL
9C74 29       0360         ADD  HL,HL
9C75 2B       0370         DEC  HL
9C76 4D       0380         LD   C,L
9C77 44       0390         LD   B,H
9C78 210000   0400         LD   HL,0
9C7B 1E19     0410         LD   E,25
9C7D F5       0420 NXTHOR  PUSH AF
9C7E D5       0430         PUSH DE
9C7F C5       0440         PUSH BC
9C80 E5       0450         PUSH HL
9C81 110000   0460         LD   DE,0
9C84 CD5FBC   0470         CALL SCRHOR
9C87 E1       0480         POP  HL
9C88 0608     0490         LD   B,8
9C8A 23       0500 MOVEY   INC  HL
9C8B 10FD     0510         DJNZ MOVEY
9C8D C1       0520         POP  BC
9C8E D1       0530         POP  DE
9C8F F1       0540         POP  AF
9C90 1D       0550         DEC  E
9C91 20EA     0560         JR   NZ,NXTHOR
9C93 C9       0570         RET
              0580         END
```

Graphic Explosion

If you have ever written a few action games, then I bet you have often wished for a good, fast, eye-catching explosion routine. I know that before this was written, I had tried many attempts through BASIC but all lacked the smoothness and quality of Machine Code.

This routine is followed by a good many parameters, as is shown below:

```
CALL 40000, L, R, T, B, TY, D
```

The first four parameters specify the coordinates of the box within which the explosion will occur. You will find that you will usually need a reasonably-sized box to give a good explosive effect. L and R stand for the Left and Right screen values of the box with T and B standing for Top and Bottom. The box is thus defined in a similar way to the Amstrad's WINDOW command.

There are two types of explosion available and these are selectable via the TY parameter. '0' selects the explosion that will cover the whole confines of the box. '1' selects the other type of explosion which only explodes any characters within the box. If, for example, you only had a spaceship made up of two characters within a 3 by 3 box then with explosion type 1, only those two characters would explode; with type 0, however, all 9 characters within the 3 by 3 box would explode.

The last parameter is the delay value. Smaller box areas explode faster than larger areas, so the delay rate should be adjusted accordingly. You need to experiment with the delay value to find a speed that is suitable for your program and the size of the box. The larger the actual value, the larger the delay and slower the overall explosion.

The routine is not effective for very large areas but does work in all three screen modes. It uses all the colours that are available in the mode that it is in to create the explosion.

```
1 '                EXPLOSION
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40226
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
50 DATA FE,06,C0,CD,06,B9,DD,6E,06,DD
60 DATA 66,0A,2D,25,DD,7E,08,94,4F,CD
70 DATA 1A,BC,AF,81,10,FD,47,DD,7E,04
80 DATA DD,4E,06,0D,91,4F,EB,21,00,00
90 DATA DD,7E,02,B7,20,45,C5,D5,3E,08
100 DATA F5,C5,D5,7E,23,B6,23,CB,74,28
110 DATA 03,21,00,00,12,1C,20,0A,14,7A
120 DATA E6,07,20,04,7A,D6,08,57,10,E5
130 DATA D1,7A,C6,08,57,C1,F1,3D,20,D8
140 DATA 7A,D6,40,57,7B,C6,50,5F,30,0A
150 DATA 14,7A,E6,07,20,04,7A,D6,08,57
160 DATA 0D,20,BF,D1,C1,3E,0F,F5,C5,D5
170 DATA DD,7E,00,3C,76,3D,20,FC,3E,08
180 DATA F5,C5,D5,7E,23,B6,23,CB,74,28
190 DATA 03,21,00,00,EB,A6,EB,12,1C,20
200 DATA 0A,14,7A,E6,07,20,04,7A,D6,08
210 DATA 57,10,E2,D1,7A,C6,08,57,C1,F1
220 DATA 3D,20,D5,7A,D6,40,57,7B,C6,50
230 DATA 5F,30,0A,14,7A,E6,07,20,04,7A
240 DATA D6,08,57,0D,20,BC,D1,C1,F1,3D
250 DATA 20,AB,3E,00,DD,5E,04,DD,6E,06
260 DATA DD,56,08,DD,66,0A,1D,2D,15,25
270 DATA CD,44,BC,CD,09,B9,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE
                 0001 ;      GRPH EXPLOSION
                 0002 ;
9C40             0010        ORG  40000
BC1A             0020 CHRPOS DEFL 0BC1AH
B906             0030 ROMENA DEFL 0B906H
B909             0040 ROMDIS DEFL 0B909H
BC44             0050 FILBOX DEFL 0BC44H
9C40 FE06        0060        CP   6
9C42 C0          0070        RET  NZ
9C43 CD06B9      0080        CALL ROMENA
9C46 DD6E06      0090        LD   L,(IX+6)
9C49 DD660A      0100        LD   H,(IX+10)
9C4C 2D          0110        DEC  L
9C4D 25          0120        DEC  H
9C4E DD7E08      0130        LD   A,(IX+8)
9C51 94          0140        SUB  H
9C52 4F          0150        LD   C,A
9C53 CD1ABC      0160        CALL CHRPOS
9C56 AF          0170        XOR  A
9C57 81          0180 BYTES  ADD  A,C
9C58 10FD        0190        DJNZ BYTES
9C5A 47          0200        LD   B,A
9C5B DD7E04      0210        LD   A,(IX+4)
9C5E DD4E06      0220        LD   C,(IX+6)
9C61 0D          0230        DEC  C
9C62 91          0240        SUB  C
9C63 4F          0250        LD   C,A
9C64 EB          0260        EX   DE,HL
9C65 210000      0270        LD   HL,0
9C68 DD7E02      0280        LD   A,(IX+2)
9C6B B7          0290        OR   A
9C6C 2045        0300        JR   NZ,PART2
9C6E C5          0310        PUSH BC
```

```
9C6F D5         0320          PUSH DE
9C70 3E08       0330 NXTROW  LD   A,8
9C72 F5         0340 NXTLIN  PUSH AF
9C73 C5         0350          PUSH BC
9C74 D5         0360          PUSH DE
9C75 7E         0370 NXTBYT  LD   A,(HL)
9C76 23         0380          INC  HL
9C77 B6         0390          OR   (HL)
9C78 23         0400          INC  HL
9C79 CB74       0410          BIT  6,H
9C7B 2803       0420          JR   Z,INROM
9C7D 210000     0430          LD   HL,0
9C80 12         0440 INROM   LD   (DE),A
9C81 1C         0450          INC  E
9C82 200A       0460          JR   NZ,GOTBYT
9C84 14         0470          INC  D
9C85 7A         0480          LD   A,D
9C86 E607       0490          AND  7
9C88 2004       0500          JR   NZ,GOTBYT
9C8A 7A         0510          LD   A,D
9C8B D608       0520          SUB  8
9C8D 57         0530          LD   D,A
9C8E 10E5       0540 GOTBYT  DJNZ NXTBYT
9C90 D1         0550          POP  DE
9C91 7A         0560          LD   A,D
9C92 C608       0570          ADD  A,8
9C94 57         0580          LD   D,A
9C95 C1         0590          POP  BC
9C96 F1         0600          POP  AF
9C97 3D         0610          DEC  A
9C98 20D8       0620          JR   NZ,NXTLIN
9C9A 7A         0630          LD   A,D
9C9B D640       0640          SUB  64
9C9D 57         0650          LD   D,A
9C9E 7B         0660          LD   A,E
9C9F C650       0670          ADD  A,80
```

```
9CA1 5F         0680        LD   E,A
9CA2 300A       0690        JR   NC,GOTROW
9CA4 14         0700        INC  D
9CA5 7A         0710        LD   A,D
9CA6 E607       0720        AND  7
9CA8 2004       0730        JR   NZ,GOTROW
9CAA 7A         0740        LD   A,D
9CAB D608       0750        SUB  8
9CAD 57         0760        LD   D,A
9CAE 0D         0770 GOTROW DEC  C
9CAF 20BF       0780        JR   NZ,NXTROW
9CB1 D1         0790        POP  DE
9CB2 C1         0800        POP  BC
9CB3 3E0F       0810 PART2  LD   A,15
9CB5 F5         0820 FADE   PUSH AF
9CB6 C5         0830        PUSH BC
9CB7 D5         0840        PUSH DE
9CB8 DD7E00     0850        LD   A,(IX+0)
9CBB 3C         0860        INC  A
9CBC 76         0870 PAUSE  HALT
9CBD 3D         0880        DEC  A
9CBE 20FC       0890        JR   NZ,PAUSE
9CC0 3E08       0900 NEWROW LD   A,8
9CC2 F5         0910 NEWLIN PUSH AF
9CC3 C5         0920        PUSH BC
9CC4 D5         0930        PUSH DE
9CC5 7E         0940 NEWBYT LD   A,(HL)
9CC6 23         0950        INC  HL
9CC7 B6         0960        OR   (HL)
9CC8 23         0970        INC  HL
9CC9 CB74       0980        BIT  6,H
9CCB 2803       0990        JR   Z,ROM
9CCD 210000     1000        LD   HL,0
9CD0 EB         1010 ROM    EX   DE,HL
9CD1 A6         1020        AND  (HL)
9CD2 EB         1030        EX   DE,HL
```

```
9CD3 12         1040            LD    (DE),A
9CD4 1C         1050            INC   E
9CD5 200A       1060            JR    NZ,FNDBYT
9CD7 14         1070            INC   D
9CD8 7A         1080            LD    A,D
9CD9 E607       1090            AND   7
9CDB 2004       1100            JR    NZ,FNDBYT
9CDD 7A         1110            LD    A,D
9CDE D608       1120            SUB   8
9CE0 57         1130            LD    D,A
9CE1 10E2       1140 FNDBYT DJNZ  NEWBYT
9CE3 D1         1150            POP   DE
9CE4 7A         1160            LD    A,D
9CE5 C608       1170            ADD   A,8
9CE7 57         1180            LD    D,A
9CE8 C1         1190            POP   BC
9CE9 F1         1200            POP   AF
9CEA 3D         1210            DEC   A
9CEB 20D5       1220            JR    NZ,NEWLIN
9CED 7A         1230            LD    A,D
9CEE D640       1240            SUB   64
9CF0 57         1250            LD    D,A
9CF1 7B         1260            LD    A,E
9CF2 C650       1270            ADD   A,80
9CF4 5F         1280            LD    E,A
9CF5 300A       1290            JR    NC,FNDROW
9CF7 14         1300            INC   D
9CF8 7A         1310            LD    A,D
9CF9 E607       1320            AND   7
9CFB 2004       1330            JR    NZ,FNDROW
9CFD 7A         1340            LD    A,D
9CFE D608       1350            SUB   8
9D00 57         1360            LD    D,A
9D01 0D         1370 FNDROW DEC   C
9D02 20BC       1380            JR    NZ,NEWROW
9D04 D1         1390            POP   DE
```

```
9D05 C1         1400         POP   BC
9D06 F1         1410         POP   AF
9D07 3D         1420         DEC   A
9D08 20AB       1430         JR    NZ,FADE
9D0A 3E00       1440         LD    A,0
9D0C DD5E04     1450         LD    E,(IX+4)
9D0F DD6E06     1460         LD    L,(IX+6)
9D12 DD5608     1470         LD    D,(IX+8)
9D15 DD660A     1480         LD    H,(IX+10)
9D18 1D         1490         DEC   E
9D19 2D         1500         DEC   L
9D1A 15         1510         DEC   D
9D1B 25         1520         DEC   H
9D1C CD44BC     1530         CALL  FILBOX
9D1F CD09B9     1540         CALL  ROMDIS
9D22 C9         1550         RET
                1560         END
```

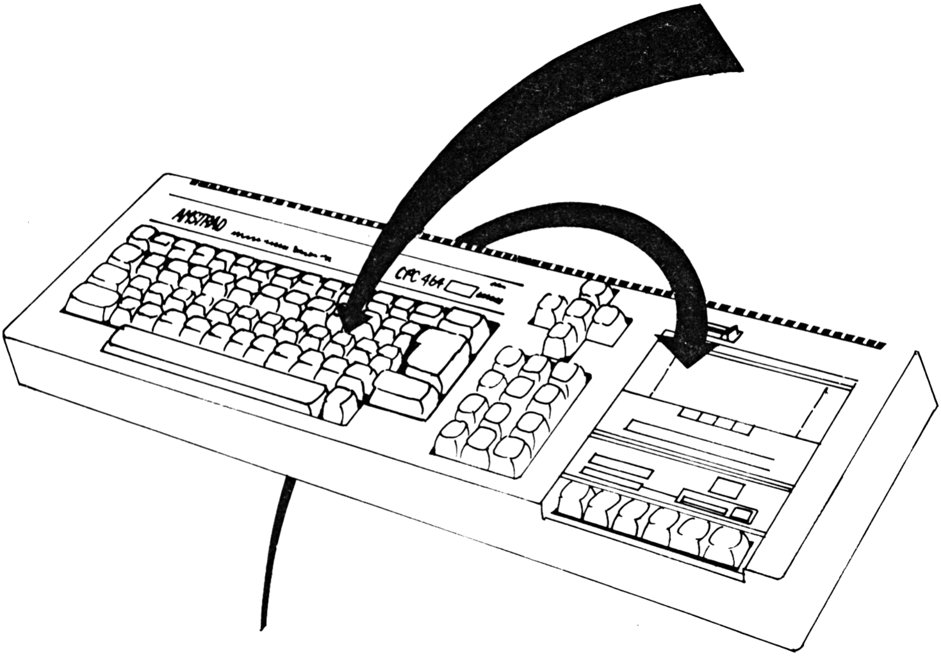
Screen Checker

This routine performs a most useful function; it checks what character is situated at a particular screen location. This function is one often used in moving graphics games and many home computers have a BASIC keyword – such as SCRN on the Spectrum – to do the job of our routine.

The routine works with all three of the Amstrad's screen modes and should be called in the format shown below:

C% = 0: CALL 40000, A, B, @C%

A and B are the coordinates of the screen position to be checked while C% is the variable which will contain the ASCII code of the character resident at the specified screen position. C% must be an integer (hence the '%' sign after the variable name) and needs to be defined before the CALL otherwise an error is generated.



```
1 '              SCREEN CHECKER
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40019
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
50 DATA DD,6E,02,DD,66,04,CD,75,BB,CD
60 DATA 60,BB,DD,6E,00,DD,66,01,77,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

              0001 ;      SCRN CHECKER
              0002 ;
9C40          0010        ORG  40000
BB75          0020 CURSOR DEFL 0BB75H
BB60          0030 RDCHAR DEFL 0BB60H
9C40 DD6E02   0040        LD   L,(IX+2)
9C43 DD6604   0050        LD   H,(IX+4)
9C46 CD75BB   0060        CALL CURSOR
9C49 CD60BB   0070        CALL RDCHAR
9C4C DD6E00   0080        LD   L,(IX+0)
9C4F DD6601   0090        LD   H,(IX+1)
9C52 77       0100        LD   (HL),A
9C53 C9       0110        RET
              0120        END
```

Nine Character Print

A previous routine, Four Character Print, has been redesigned and enlarged to provide a 3 by 3 block of characters that can be printed to any position on the graphics screen without the use of TAG and TAGOFF or any other complex commands.

The principles are exactly the same for this routine as for the previous one. CALL 40000, X, Y, C will start the printing of the block from position X and Y on the graphics screen, with that coordinate corresponding to the top left corner of the block.

Of course, the major difference is the number of characters printed. C still equals the ASCII code of the first character to be printed but now the next 8 characters are also printed. This simply calls for a little more thinking and design on your part.

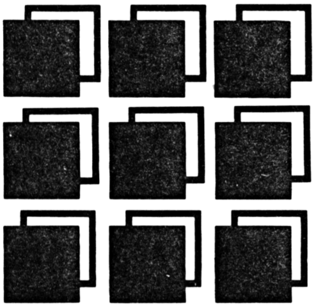
As with the previous routine, the main area of use for this routine is in moving chunks of UDGs around the screen with ease.

```
1 '             9 CHAR PRINT
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40055
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
50 DATA DD,4E,00,DD,6E,02,DD,66,03,DD
60 DATA 5E,04,DD,56,05,06,03,D5,E5,C5
70 DATA CD,C0,BB,C1,79,0C,C5,CD,FC,BB
80 DATA C1,79,0C,C5,CD,FC,BB,C1,79,0C
90 DATA C5,CD,FC,BB,C1,E1,D1,3E,10,2B
100 DATA 3D,20,FC,10,DA,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

              0001 ;      9CH  PRINT
              0002 ;
9C40          0010        ORG  40000
BBC6          0020 ASKCUR DEFL 0BBC6H
BBC0          0030 MOVABS DEFL 0BBC0H
BBFC          0040 GRACHR DEFL 0BBFCH
9C40 DD4E00   0050        LD   C,(IX+0)
9C43 DD6E02   0060        LD   L,(IX+2)
9C46 DD6603   0070        LD   H,(IX+3)
9C49 DD5E04   0080        LD   E,(IX+4)
9C4C DD5605   0090        LD   D,(IX+5)
9C4F 0603     0100        LD   B,3
9C51 D5       0110 NXTCHR PUSH DE
9C52 E5       0120        PUSH HL
9C53 C5       0130        PUSH BC
9C54 CDC0BB   0140        CALL MOVABS
9C57 C1       0150        POP  BC
9C58 79       0160        LD   A,C
9C59 0C       0170        INC  C
9C5A C5       0180        PUSH BC
9C5B CDFCBB   0190        CALL GRACHR
9C5E C1       0200        POP  BC
9C5F 79       0210        LD   A,C
9C60 0C       0220        INC  C
9C61 C5       0230        PUSH BC
9C62 CDFCBB   0240        CALL GRACHR
9C65 C1       0250        POP  BC
9C66 79       0260        LD   A,C
9C67 0C       0270        INC  C
9C68 C5       0280        PUSH BC
9C69 CDFCBB   0290        CALL GRACHR
9C6C C1       0300        POP  BC
```

```
9C6D E1         0310          POP   HL
9C6E D1         0320          POP   DE
9C6F 3E10       0330          LD    A,16
9C71 2B         0340 NEWROW   DEC   HL
9C72 3D         0350          DEC   A
9C73 20FC       0360          JR    NZ,NEWROW
9C75 10DA       0370          DJNZ  NXTCHR
9C77 C9         0380          RET
                0390          END
```



This little demonstration routine bounces a ball around the screen and shows the speed of the nine character print routine. Remember when watching the ball that it is moving across the graphics and not the text screen.

```
10 SYMBOL 240,0,0,0,0,0,0,0,0
20 SYMBOL 241,0,0,0,0,0,0,0,0
30 SYMBOL 242,0,0,0,0,0,0,0,0
40 SYMBOL 243,0,0,0,0,0,0,0,0
50 SYMBOL 244,60,126,255,255,255,255,126
,60
60 SYMBOL 245,0,0,0,0,0,0,0,0
70 SYMBOL 246,0,0,0,0,0,0,0,0
```

```
80 SYMBOL 247,0,0,0,0,0,0,0,0
90 SYMBOL 248,0,0,0,0,0,0,0,0
100 BORDER 2:CLS
110 x=320:y=200:a=16:b=16
120 CALL 40000,x,y,240
130 x=x+a:y=y+b
140 IF x<10 OR x>600 THEN a=-a:SOUND 1,4
0,5,7
150 IF y<50 OR y>390 THEN b=-b:SOUND 1,5
0,5,7
160 GOTO 120
```

Italics

It is possible to create a complete new character set with the SYMBOL command and thus a character set could be defined that is in italics. However, consider the amount of work that would go into designing all the capital letters, small letters, numbers and symbols to make up a complete set on an 8 by 8 grid, and I'm sure you will agree that this program is a lot easier and quicker to use.

The format is:

```
CALL 40000, R
```

R is the row which you want turned into italic characters. We did originally consider turning the whole screen into italics with no row parameters but felt that this would limit the routine's use and flexibility. We also thought about having to specify the row and column, thus italicising just one character, but felt that for a sentence or phrase this would take a very large and time-consuming number of calls. The resultant parameter – that of the specified row – is what we consider to be the best compromise and is the sort of situation that you will come across when you write your own Machine Code routines.

Unlike most of our routines, this one only works in screen mode 1. This is because the routine accesses the screen directly. The characters are sloped to the right, as is the case with italicised characters, and in doing so a small part of the 40th column character slips off the screen.

```
1 '                  ITALICS
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40071
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
50 DATA B7,C8,DD,6E,00,2D,26,27,CD,1A
60 DATA BC,23,0E,07,C5,E5,CB,3E,06,4F
70 DATA 5D,54,7D,2D,B7,20,0A,7C,25,E6
80 DATA 07,20,04,7C,C6,08,67,1A,CB,3E
90 DATA 38,04,CB,9F,18,02,CB,DF,CB,5E
100 DATA 28,02,CB,FF,12,10,DB,CB,9E,E1
110 DATA 0D,20,D0,7C,C6,08,67,C1,0D,20
120 DATA C7,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

                  0001 ;               ITALICS
                  0002 ;
9C40              0010         ORG   40000
BC1A              0020 CHRPOS  DEFL  0BC1AH
9C40 B7           0030         OR    A
9C41 C8           0040         RET   Z
9C42 DD6E00       0050         LD    L,(IX+0)
9C45 2D           0060         DEC   L
9C46 2627         0070         LD    H,39
9C48 CD1ABC       0080         CALL  CHRPOS
9C4B 23           0090         INC   HL
9C4C 0E07         0100         LD    C,7
9C4E C5           0110 NXTROW  PUSH  BC
9C4F E5           0120 SCROLL  PUSH  HL
9C50 CB3E         0130         SRL   (HL)
```

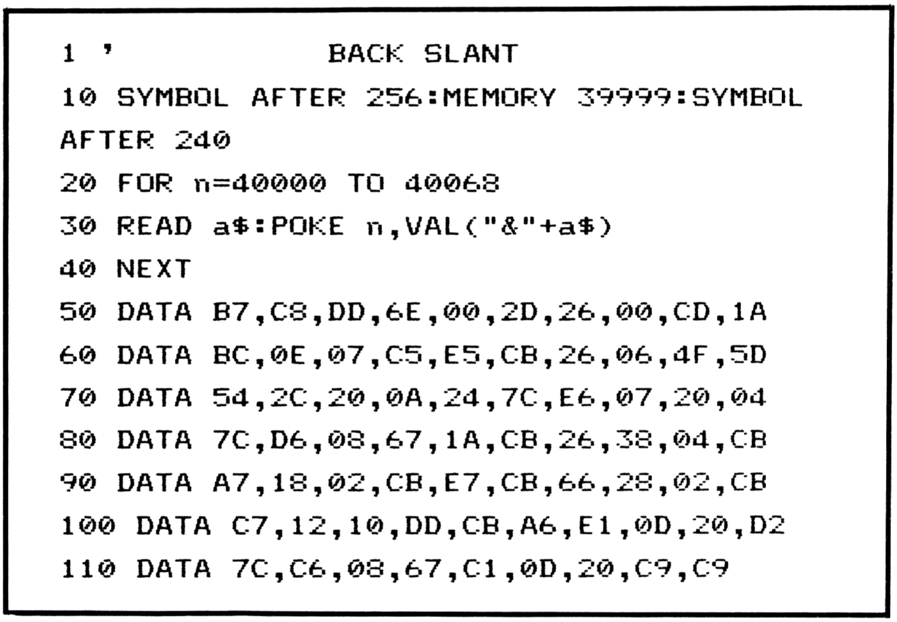
```
9C52 064F    0140         LD   B,79
9C54 5D      0150 NXTBYT  LD   E,L
9C55 54      0160         LD   D,H
9C56 7D      0170         LD   A,L
9C57 2D      0180         DEC  L
9C58 B7      0190         OR   A
9C59 200A    0200         JR   NZ,HLOK
9C5B 7C      0210         LD   A,H
9C5C 25      0220         DEC  H
9C5D E607    0230         AND  7
9C5F 2004    0240         JR   NZ,HLOK
9C61 7C      0250         LD   A,H
9C62 C608    0260         ADD  A,8
9C64 67      0270         LD   H,A
9C65 1A      0280 HLOK    LD   A,(DE)
9C66 CB3E    0290         SRL  (HL)
9C68 3804    0300         JR   C,SETBIT
9C6A CB9F    0310         RES  3,A
9C6C 1802    0320         JR   TSTBIT
9C6E CBDF    0330 SETBIT  SET  3,A
9C70 CB5E    0340 TSTBIT  BIT  3,(HL)
9C72 2802    0350         JR   Z,BITOK
9C74 CBFF    0360         SET  7,A
9C76 12      0370 BITOK   LD   (DE),A
9C77 10DB    0380         DJNZ NXTBYT
9C79 CB9E    0390         RES  3,(HL)
9C7B E1      0400         POP  HL
9C7C 0D      0410         DEC  C
9C7D 20D0    0420         JR   NZ,SCROLL
9C7F 7C      0430         LD   A,H
9C80 C608    0440         ADD  A,8
9C82 67      0450         LD   H,A
9C83 C1      0460         POP  BC
9C84 0D      0470         DEC  C
9C85 20C7    0480         JR   NZ,NXTROW
9C87 C9      0490         RET
             0500         END
```

Back Slant

This routine has much in common with the Italics routine earlier in the book. It performs a backwards slant of all the characters on the specified row. The CALL statement is exactly the same as for the Italics routine:

CALL 40000, R

The only difference is that it is the 1st column character, rather than the 40th column character, which is partly lost.

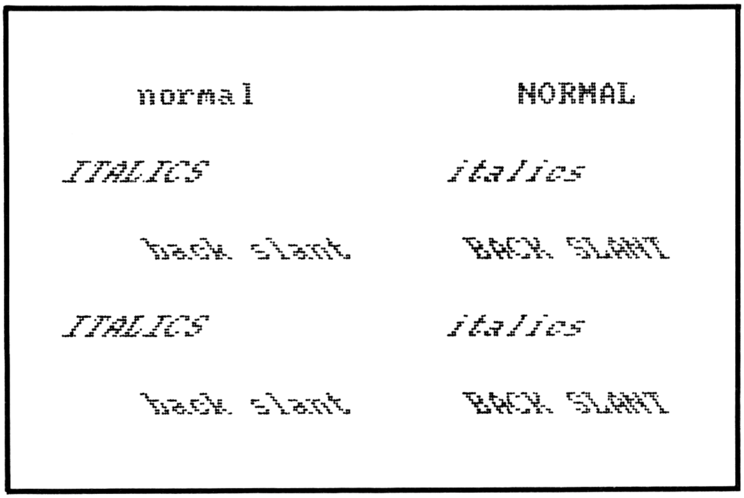


```
1 '              BACK SLANT
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40068
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
50 DATA B7,C8,DD,6E,00,2D,26,00,CD,1A
60 DATA BC,0E,07,C5,E5,CB,26,06,4F,5D
70 DATA 54,2C,20,0A,24,7C,E6,07,20,04
80 DATA 7C,D6,08,67,1A,CB,26,38,04,CB
90 DATA A7,18,02,CB,E7,CB,66,28,02,CB
100 DATA C7,12,10,DD,CB,A6,E1,0D,20,D2
110 DATA 7C,C6,08,67,C1,0D,20,C9,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

                    0001 ;                BACKSLANT
                    0002 ;
9C40                0010            ORG   40000
BC1A                0020 CHRPOS DEFL  0BC1AH
9C40 B7             0030            OR    A
9C41 C8             0040            RET   Z
9C42 DD6E00         0050            LD    L,(IX+0)
9C45 2D             0060            DEC   L
9C46 2600           0070            LD    H,0
9C48 CD1ABC         0080            CALL  CHRPOS
9C4B 0E07           0090            LD    C,7
9C4D C5             0100 NXTROW PUSH  BC
9C4E E5             0110 SCROLL PUSH  HL
9C4F CB26           0120            SLA   (HL)
9C51 064F           0130            LD    B,79
9C53 5D             0140 NXTBYT LD    E,L
9C54 54             0150            LD    D,H
9C55 2C             0160            INC   L
9C56 200A           0170            JR    NZ,HLOK
9C58 24             0180            INC   H
9C59 7C             0190            LD    A,H
9C5A E607           0200            AND   7
9C5C 2004           0210            JR    NZ,HLOK
9C5E 7C             0220            LD    A,H
9C5F D608           0230            SUB   8
9C61 67             0240            LD    H,A
9C62 1A             0250 HLOK   LD    A,(DE)
9C63 CB26           0260            SLA   (HL)
9C65 3804           0270            JR    C,SETBIT
9C67 CBA7           0280            RES   4,A
9C69 1802           0290            JR    TSTBIT
9C6B CBE7           0300 SETBIT SET   4,A
```

```
9C6D CB66       0310 TSTBIT BIT  4,(HL)
9C6F 2802       0320        JR   Z,BITOK
9C71 CBC7       0330        SET  0,A
9C73 12         0340 BITOK  LD   (DE),A
9C74 10DD       0350        DJNZ NXTBYT
9C76 CBA6       0360        RES  4,(HL)
9C78 E1         0370        POP  HL
9C79 0D         0380        DEC  C
9C7A 20D2       0390        JR   NZ,SCROLL
9C7C 7C         0400        LD   A,H
9C7D C608       0410        ADD  A,8
9C7F 67         0420        LD   H,A
9C80 C1         0430        POP  BC
9C81 0D         0440        DEC  C
9C82 20C9       0450        JR   NZ,NXTROW
9C84 C9         0460        RET
                0470        END
```



```
    normal              NORMAL

ITALICS              italics

    back slant       BACK SLANT

ITALICS              italics

    back slant       BACK SLANT
```

Intelligent Copy

No, it doesn't have a degree or any O levels but even so it is far cleverer than the dumb copying routines there are around! This routine will copy a chunk of memory from one area to another. The format for this is:

CALL 40000, A1, A2, L

A1 is the start address of the block to be moved while L equals its length. A2 is the new start address where it is to be copied to. The length is in bytes and an area of memory from 40000 to 40240, for example, has a length of 241 bytes.

'Intelligent' as used in the title is not our choice of adjective but rather a term describing a certain routine. A dumb copy will simply move one area of memory from one location to another without making any checks; an intelligent copy allows for overlapping block addresses.

Suppose, for example, that your specified piece of memory starts at 30000 and continues for 10 bytes but that you wish to relocate it at 30005, thus making it overlap. In this situation a dumb copy would simply copy straight over, losing the contents of half the memory area, whereas an intelligent copy checks for this occurrence and copies the new material into its new memory area without overwriting any part of it.

```
1 '           INTELLIGENT COPY
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 FOR n=40000 TO 40039
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
50 DATA FE,03,C0,DD,4E,00,DD,46,01,DD
60 DATA 5E,02,DD,56,03,DD,6E,04,DD,66
70 DATA 05,B7,E5,ED,52,E1,38,03,ED,B0
80 DATA C9,09,EB,09,EB,2B,1B,ED,B8,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

              0001 ;       INTL COPY
              0002 ;
9C40          0010         ORG  40000
9C40 FE03     0020         CP   3
9C42 C0       0030         RET  NZ
9C43 DD4E00   0040         LD   C,(IX+0)
9C46 DD4601   0050         LD   B,(IX+1)
9C49 DD5E02   0060         LD   E,(IX+2)
9C4C DD5603   0070         LD   D,(IX+3)
9C4F DD6E04   0080         LD   L,(IX+4)
9C52 DD6605   0090         LD   H,(IX+5)
9C55 B7       0100         OR   A
9C56 E5       0110         PUSH HL
9C57 ED52     0120         SBC  HL,DE
9C59 E1       0130         POP  HL
9C5A 3803     0140         JR   C,LDDEC
9C5C EDB0     0150         LDIR
9C5E C9       0160         RET
9C5F 09       0170 LDDEC   ADD  HL,BC
9C60 EB       0180         EX   DE,HL
9C61 09       0190         ADD  HL,BC
9C62 EB       0200         EX   DE,HL
9C63 2B       0210         DEC  HL
9C64 1B       0220         DEC  DE
9C65 EDB8     0230         LDDR
9C67 C9       0240         RET
              0250         END
```

Screen Mover

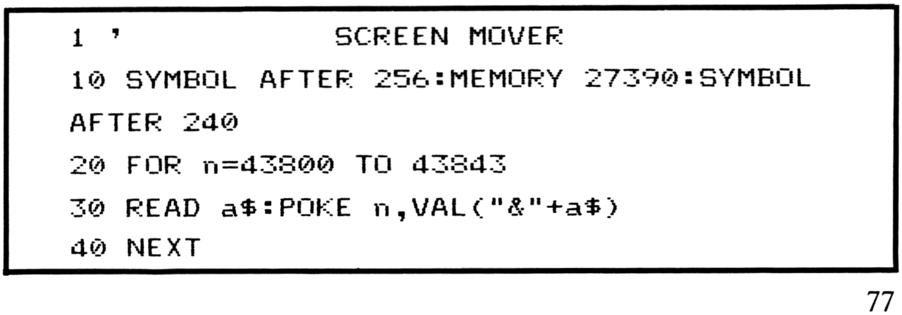
This routine takes a screen (worth about 16K in memory terms) and stores it elsewhere in memory. It can then be called back to be displayed on screen later.

Let's first discuss how to use it. To save the screen memory and transfer it to another memory area (starting, in fact, at address 27392) one should type CALL 43800. To recall that screen display, type CALL 43812 – everything on the screen will be erased and replaced with the stored screen display.

The routine offers an additional feature. If you CALL 43824, the stored screen will be displayed on top of the present screen without erasing the present screen. Obviously, a screen position occupied by a character on both screens will appear as a bit of a mess. However, blank areas of one screen display filled by the other display can be used most effectively in a number of programs from adventure games to educational and business packages.

Looking at the BASIC listing, you may notice that instead of our normal MEMORY 39999 command (reserving a couple of thousand bytes for our Machine Code routines starting at 40000) this routine reserves memory from 27390 upwards. This is, of course, to allow for the screen to be stored and this MEMORY 27390 command must be included if you use this routine in one of your own programs.

There is another point to bear in mind when using this program. It is necessary to perform a MODE command before CALLing the routine since, if the screen has been scrolled or altered particularly, one will find that the re-displayed screen will start in the wrong place. Obviously, whatever mode you are actually in will determine what exact mode command you perform. A routine working in mode 2 should have a MODE 2 command before the CALL.



```
1 '              SCREEN MOVER
10 SYMBOL AFTER 256:MEMORY 27390:SYMBOL
AFTER 240
20 FOR n=43800 TO 43843
30 READ a$:POKE n,VAL("&"+a$)
40 NEXT
```

```
50 DATA 01,00,40,11,00,6B,21,00,C0,ED
60 DATA B0,C9,01,00,40,11,00,C0,21,00
70 DATA 6B,ED,B0,C9,01,00,40,11,00,C0
80 DATA 21,00,6B,1A,AE,12,13,23,0B,78
90 DATA B1,20,F6,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE
               0001 ;         SCRN MOVER
               0002 ;
AB18           0010           ORG  43800
AB18 010040    0020           LD   BC,4000H
AB1B 11006B    0030           LD   DE,6B00H
AB1E 2100C0    0040           LD   HL,0C000H
AB21 EDB0      0050           LDIR
AB23 C9        0060           RET
AB24 010040    0070           LD   BC,4000H
AB27 1100C0    0080           LD   DE,0C000H
AB2A 21006B    0090           LD   HL,6B00H
AB2D EDB0      0100           LDIR
AB2F C9        0110           RET
AB30 010040    0120           LD   BC,4000H
AB33 1100C0    0130           LD   DE,0C000H
AB36 21006B    0140           LD   HL,6B00H
AB39 1A        0150 LOOP      LD   A,(DE)
AB3A AE        0160           XOR  (HL)
AB3B 12        0170           LD   (DE),A
AB3C 13        0180           INC  DE
AB3D 23        0190           INC  HL
AB3E 0B        0200           DEC  BC
AB3F 78        0210           LD   A,B
AB40 B1        0220           OR   C
AB41 20F6      0230           JR   NZ,LOOP
AB43 C9        0240           RET
               0250           END
```

Graphics Package

This package has been created by taking a number of the graphics routines in this book and re-writing them, therefore allowing them to be used in one large routine. This single routine also creates new BASIC commands, one for each new graphics feature.

This single routine can be entered into memory, then allowing the programmer to use any of the new BASIC keywords within their own BASIC program.

Typing RUN after entering the program into memory will provide you with these new commands:

¦INVERT	**Character Invert**
¦EXPLODE	**Graphical Explosion**
¦WRAPLEFT	**Left Scroll**
¦WRAPRIGHT	**Right Scroll**
¦PRINTFOUR	**Four Graphic Print**

These new BASIC keywords take the place of CALL 40000's but must of course still be followed by the various parameters. To check up on these, and to get a better idea of how each individual command works, look up the appropriate routines throughout the book.

This package can be used for improving the display of a program, be it a game, an educational program or even (perhaps with the exception of ¦ EXPLODE) a business package.

```
1 '             GRAPHICS PACKAGE
10 SYMBOL AFTER 256:MEMORY 39999:SYMBOL
AFTER 240
20 DIM x(6)
30 FOR c=0 TO 6:READ a$:x(c)=VAL("&"+a$)
:NEXT
40 c=0:sum=0
50 FOR n=40000 TO 40609
60 READ a$:v=VAL("&"+a$)
```

```
70 sum=sum+v:POKE n,v
80 IF (n+1-40000) MOD 100<>0 GOTO 110
90 IF sum<>x(c) THEN PRINT"*ERROR IN DAT
A*";CHR$(7):PRINT"Check lines";200+100*c
;"to";290+100*c:END
100 sum=0:c=c+1
110 NEXT
120 IF sum<>x(c) THEN PRINT"*ERROR IN DA
TA*";CHR$(7):PRINT"Check line 800"
130 '
140 '            CHECKSUMS
150 '
160 DATA 2BCA,2EC6,2831,2803,2C6B,3029,0
45D
170 '
180 '           MACHINE CODE
190 '
200 DATA 01,4A,9C,21,5B,9C,CD,D1,BC,C9
210 DATA 5F,9C,C3,87,9C,C3,D7,9C,C3,BA
220 DATA 9D,C3,12,9E,C3,71,9E,00,00,00
230 DATA 00,49,4E,56,45,52,D4,45,58,50
240 DATA 4C,4F,44,C5,57,52,41,50,4C,45
250 DATA 46,D4,57,52,41,50,52,49,47,48
260 DATA D4,50,52,49,4E,54,46,4F,55,D2
270 DATA 00,FE,03,C0,DD,6E,00,DD,66,01
280 DATA 7E,B7,C8,23,5E,23,56,DD,6E,02
290 DATA DD,66,04,47,C5,E5,CD,75,BB,1A
300 DATA CD,AE,9C,13,E1,25,C1,10,F1,C9
310 DATA D5,CD,A5,BB,EB,CD,AE,BB,06,07
320 DATA 23,10,FD,F5,CD,06,B9,0E,08,1A
330 DATA 06,08,1F,CB,16,10,FB,13,2B,0D
340 DATA 20,F3,CD,09,B9,F1,CD,5A,BB,D1
350 DATA C9,FE,06,C0,CD,06,B9,DD,6E,06
360 DATA DD,66,0A,2D,25,DD,7E,08,94,4F
370 DATA CD,1A,BC,AF,81,10,FD,47,DD,7E
```

```
380 DATA 04,DD,4E,06,0D,91,4F,EB,21,00
390 DATA 00,DD,7E,02,B7,20,45,C5,D5,3E
400 DATA 08,F5,C5,D5,7E,23,B6,23,CB,74
410 DATA 28,03,21,00,00,12,1C,20,0A,14
420 DATA 7A,E6,07,20,04,7A,D6,08,57,10
430 DATA E5,D1,7A,C6,08,57,C1,F1,3D,20
440 DATA D8,7A,D6,40,57,7B,C6,50,5F,30
450 DATA 0A,14,7A,E6,07,20,04,7A,D6,08
460 DATA 57,0D,20,BF,D1,C1,3E,0F,F5,C5
470 DATA D5,DD,7E,00,3C,76,3D,20,FC,3E
480 DATA 08,F5,C5,D5,7E,23,B6,23,CB,74
490 DATA 28,03,21,00,00,EB,A6,EB,12,1C
500 DATA 20,0A,14,7A,E6,07,20,04,7A,D6
510 DATA 08,57,10,E2,D1,7A,C6,08,57,C1
520 DATA F1,3D,20,D5,7A,D6,40,57,7B,C6
530 DATA 50,5F,30,0A,14,7A,E6,07,20,04
540 DATA 7A,D6,08,57,0D,20,BC,D1,C1,F1
550 DATA 3D,20,AB,3E,00,DD,5E,04,DD,6E
560 DATA 06,DD,56,08,DD,66,0A,1D,2D,15
570 DATA 25,CD,44,BC,CD,09,B9,C9,FE,03
580 DATA C0,DD,6E,02,2D,DD,66,04,25,CD
590 DATA 1A,BC,DD,46,00,CB,20,05,0E,08
600 DATA E5,C5,CB,26,7E,F5,5D,54,2C,20
610 DATA 0A,24,7C,E6,07,20,04,7C,D6,08
620 DATA 67,1A,CB,26,38,04,CB,A7,18,02
630 DATA CB,E7,CB,66,28,02,CB,C7,12,10
640 DATA DD,F1,38,04,CB,A6,18,02,CB,E6
650 DATA CB,67,28,02,CB,C6,C1,E1,7C,C6
660 DATA 08,67,0D,20,BF,C9,FE,03,C0,DD
670 DATA 6E,02,2D,DD,4E,00,DD,7E,04,81
680 DATA D6,02,67,CD,1A,BC,23,41,CB,20
690 DATA 05,0E,08,E5,C5,CB,3E,7E,F5,5D
700 DATA 54,7D,2D,B7,20,0A,7C,25,E6,07
710 DATA 20,04,7C,C6,08,67,1A,CB,3E,38
720 DATA 04,CB,9F,18,02,CB,DF,CB,5E,28
```

```
730 DATA 02,CB,FF,12,10,DB,F1,38,04,CB
740 DATA 9E,18,02,CB,DE,CB,5F,28,02,CB
750 DATA FE,C1,E1,7C,C6,08,67,0D,20,BD
760 DATA C9,DD,4E,00,DD,6E,02,DD,66,03
770 DATA DD,5E,04,DD,56,05,06,02,D5,E5
780 DATA C5,CD,C0,BB,C1,79,0C,C5,CD,FC
790 DATA BB,C1,79,0C,C5,CD,FC,BB,C1,E1
800 DATA D1,3E,10,2B,3D,20,FC,10,E1,C9
```

```
Editor Assembler
AMMAS 1.1
Copyright 1985 PICTURESQUE

                 0001 ;      GR   PACKAGE
                 0002 ;
9C40             0010        ORG  40000
BCD1             0020 LOGEXT DEFL 0BCD1H
BB75             0030 CURSOR DEFL 0BB75H
BBA5             0040 MATRIX DEFL 0BBA5H
BBAE             0050 MTABLE DEFL 0BBAEH
B906             0060 ROMENA DEFL 0B906H
B909             0070 ROMDIS DEFL 0B909H
BB5A             0080 TXTOUT DEFL 0BB5AH
BC1A             0090 CHRPOS DEFL 0BC1AH
BC44             0100 FILBOX DEFL 0BC44H
BBC0             0110 MOVABS DEFL 0BBC0H
BBFC             0120 GRACHR DEFL 0BBFCH
9C40 014A9C      0130        LD   BC,TABLE
9C43 215B9C      0140        LD   HL,SPACE
9C46 CDD1BC      0150        CALL LOGEXT
9C49 C9          0160        RET
9C4A 5F9C        0170 TABLE  DEFW NAMETB
9C4C C3879C      0180        JP   INVERT
9C4F C3D79C      0190        JP   EXPLOD
```

```
9C52 C3BA9D    0200        JP   LWRAP
9C55 C3129E    0210        JP   RWRAP
9C58 C3719E    0220        JP   GRPNT4
9C5B 00        0230 SPACE  DEFB 0,0,0,0
     00 00 00
9C5F           0240 NAMETB DEFM "INVER"
9C64 D4        0250        DEFB "T"+80H
9C65           0260        DEFM "EXPLOD"
9C6B C5        0270        DEFB "E"+80H
9C6C           0280        DEFM "WRAPLEF"
9C73 D4        0290        DEFB "T"+80H
9C74           0300        DEFM "WRAPRIGH"
9C7C D4        0310        DEFB "T"+80H
9C7D           0320        DEFM "PRINTFOU"
9C85 D2        0330        DEFB "R"+80H
9C86 00        0340        DEFB 0
9C87 FE03      0350 INVERT CP   3
9C89 C0        0360        RET  NZ
9C8A DD6E00    0370        LD   L,(IX+0)
9C8D DD6601    0380        LD   H,(IX+1)
9C90 7E        0390        LD   A,(HL)
9C91 B7        0400        OR   A
9C92 C8        0410        RET  Z
9C93 23        0420        INC  HL
9C94 5E        0430        LD   E,(HL)
9C95 23        0440        INC  HL
9C96 56        0450        LD   D,(HL)
9C97 DD6E02    0460        LD   L,(IX+2)
9C9A DD6604    0470        LD   H,(IX+4)
9C9D 47        0480        LD   B,A
9C9E C5        0490 NXTCHR PUSH BC
9C9F E5        0500        PUSH HL
9CA0 CD75BB    0510        CALL CURSOR
9CA3 1A        0520        LD   A,(DE)
9CA4 CDAE9C    0530        CALL INVCHR
9CA7 13        0540        INC  DE
9CA8 E1        0550        POP  HL
```

```
9CA9 25          0560          DEC  H
9CAA C1          0570          POP  BC
9CAB 10F1        0580          DJNZ NXTCHR
9CAD C9          0590          RET
9CAE D5          0600 INVCHR   PUSH DE
9CAF CDA5BB      0610          CALL MATRIX
9CB2 EB          0620          EX   DE,HL
9CB3 CDAEBB      0630          CALL MTABLE
9CB6 0607        0640          LD   B,7
9CB8 23          0650 ADD7     INC  HL
9CB9 10FD        0660          DJNZ ADD7
9CBB F5          0670          PUSH AF
9CBC CD06B9      0680          CALL ROMENA
9CBF 0E08        0690          LD   C,8
9CC1 1A          0700 NXTBYT   LD   A,(DE)
9CC2 0608        0710          LD   B,8
9CC4 1F          0720 NXTBIT   RRA
9CC5 CB16        0730          RL   (HL)
9CC7 10FB        0740          DJNZ NXTBIT
9CC9 13          0750          INC  DE
9CCA 2B          0760          DEC  HL
9CCB 0D          0770          DEC  C
9CCC 20F3        0780          JR   NZ,NXTBYT
9CCE CD09B9      0790          CALL ROMDIS
9CD1 F1          0800          POP  AF
9CD2 CD5ABB      0810          CALL TXTOUT
9CD5 D1          0820          POP  DE
9CD6 C9          0830          RET
9CD7 FE06        0840 EXPLOD   CP   6
9CD9 C0          0850          RET  NZ
9CDA CD06B9      0860          CALL ROMENA
9CDD DD6E06      0870          LD   L,(IX+6)
9CE0 DD660A      0880          LD   H,(IX+10)
9CE3 2D          0890          DEC  L
9CE4 25          0900          DEC  H
9CE5 DD7E08      0910          LD   A,(IX+8)
```

```
9CE8 94        0920        SUB  H
9CE9 4F        0930        LD   C,A
9CEA CD1ABC    0940        CALL CHRPOS
9CED AF        0950        XOR  A
9CEE 81        0960 BYTES  ADD  A,C
9CEF 10FD      0970        DJNZ BYTES
9CF1 47        0980        LD   B,A
9CF2 DD7E04    0990        LD   A,(IX+4)
9CF5 DD4E06    1000        LD   C,(IX+6)
9CF8 0D        1010        DEC  C
9CF9 91        1020        SUB  C
9CFA 4F        1030        LD   C,A
9CFB EB        1040        EX   DE,HL
9CFC 210000    1050        LD   HL,0
9CFF DD7E02    1060        LD   A,(IX+2)
9D02 B7        1070        OR   A
9D03 2045      1080        JR   NZ,PART2
9D05 C5        1090        PUSH BC
9D06 D5        1100        PUSH DE
9D07 3E08      1110 NXTROW LD   A,8
9D09 F5        1120 NXTLIN PUSH AF
9D0A C5        1130        PUSH BC
9D0B D5        1140        PUSH DE
9D0C 7E        1150 NXTBTE LD   A,(HL)
9D0D 23        1160        INC  HL
9D0E B6        1170        OR   (HL)
9D0F 23        1180        INC  HL
9D10 CB74      1190        BIT  6,H
9D12 2803      1200        JR   Z,INROM
9D14 210000    1210        LD   HL,0
9D17 12        1220 INROM  LD   (DE),A
9D18 1C        1230        INC  E
9D19 200A      1240        JR   NZ,GOTBYT
9D1B 14        1250        INC  D
9D1C 7A        1260        LD   A,D
9D1D E607      1270        AND  7
```

```
9D1F 2004        1280          JR    NZ,GOTBYT
9D21 7A          1290          LD    A,D
9D22 D608        1300          SUB   8
9D24 57          1310          LD    D,A
9D25 10E5        1320 GOTBYT   DJNZ  NXTBTE
9D27 D1          1330          POP   DE
9D28 7A          1340          LD    A,D
9D29 C608        1350          ADD   A,8
9D2B 57          1360          LD    D,A
9D2C C1          1370          POP   BC
9D2D F1          1380          POP   AF
9D2E 3D          1390          DEC   A
9D2F 20D8        1400          JR    NZ,NXTLIN
9D31 7A          1410          LD    A,D
9D32 D640        1420          SUB   64
9D34 57          1430          LD    D,A
9D35 7B          1440          LD    A,E
9D36 C650        1450          ADD   A,80
9D38 5F          1460          LD    E,A
9D39 300A        1470          JR    NC,GOTROW
9D3B 14          1480          INC   D
9D3C 7A          1490          LD    A,D
9D3D E607        1500          AND   7
9D3F 2004        1510          JR    NZ,GOTROW
9D41 7A          1520          LD    A,D
9D42 D608        1530          SUB   8
9D44 57          1540          LD    D,A
9D45 0D          1550 GOTROW   DEC   C
9D46 20BF        1560          JR    NZ,NXTROW
9D48 D1          1570          POP   DE
9D49 C1          1580          POP   BC
9D4A 3E0F        1590 PART2    LD    A,15
9D4C F5          1600 FADE     PUSH  AF
9D4D C5          1610          PUSH  BC
9D4E D5          1620          PUSH  DE
9D4F DD7E00      1630          LD    A,(IX+0)
```

```
9D52 3C       1640        INC  A
9D53 76       1650 PAUSE  HALT
9D54 3D       1660        DEC  A
9D55 20FC     1670        JR   NZ,PAUSE
9D57 3E08     1680 NEWROW LD   A,8
9D59 F5       1690 NEWLIN PUSH AF
9D5A C5       1700        PUSH BC
9D5B D5       1710        PUSH DE
9D5C 7E       1720 NEWBYT LD   A,(HL)
9D5D 23       1730        INC  HL
9D5E B6       1740        OR   (HL)
9D5F 23       1750        INC  HL
9D60 CB74     1760        BIT  6,H
9D62 2803     1770        JR   Z,ROM
9D64 210000   1780        LD   HL,0
9D67 EB       1790 ROM    EX   DE,HL
9D68 A6       1800        AND  (HL)
9D69 EB       1810        EX   DE,HL
9D6A 12       1820        LD   (DE),A
9D6B 1C       1830        INC  E
9D6C 200A     1840        JR   NZ,FNDBYT
9D6E 14       1850        INC  D
9D6F 7A       1860        LD   A,D
9D70 E607     1870        AND  7
9D72 2004     1880        JR   NZ,FNDBYT
9D74 7A       1890        LD   A,D
9D75 D608     1900        SUB  8
9D77 57       1910        LD   D,A
9D78 10E2     1920 FNDBYT DJNZ NEWBYT
9D7A D1       1930        POP  DE
9D7B 7A       1940        LD   A,D
9D7C C608     1950        ADD  A,8
9D7E 57       1960        LD   D,A
9D7F C1       1970        POP  BC
9D80 F1       1980        POP  AF
9D81 3D       1990        DEC  A
9D82 20D5     2000        JR   NZ,NEWLIN
```

```
9D84 7A         2010            LD    A,D
9D85 D640       2020            SUB   64
9D87 57         2030            LD    D,A
9D88 7B         2040            LD    A,E
9D89 C650       2050            ADD   A,80
9D8B 5F         2060            LD    E,A
9D8C 300A       2070            JR    NC,FNDROW
9D8E 14         2080            INC   D
9D8F 7A         2090            LD    A,D
9D90 E607       2100            AND   7
9D92 2004       2110            JR    NZ,FNDROW
9D94 7A         2120            LD    A,D
9D95 D608       2130            SUB   8
9D97 57         2140            LD    D,A
9D98 0D         2150 FNDROW     DEC   C
9D99 20BC       2160            JR    NZ,NEWROW
9D9B D1         2170            POP   DE
9D9C C1         2180            POP   BC
9D9D F1         2190            POP   AF
9D9E 3D         2200            DEC   A
9D9F 20AB       2210            JR    NZ,FADE
9DA1 3E00       2220            LD    A,0
9DA3 DD5E04     2230            LD    E,(IX+4)
9DA6 DD6E06     2240            LD    L,(IX+6)
9DA9 DD5608     2250            LD    D,(IX+8)
9DAC DD660A     2260            LD    H,(IX+10)
9DAF 1D         2270            DEC   E
9DB0 2D         2280            DEC   L
9DB1 15         2290            DEC   D
9DB2 25         2300            DEC   H
9DB3 CD44BC     2310            CALL  FILBOX
9DB6 CD09B9     2320            CALL  ROMDIS
9DB9 C9         2330            RET
9DBA FE03       2340 LWRAP      CP    3
9DBC C0         2350            RET   NZ
9DBD DD6E02     2360            LD    L,(IX+2)
```

```
9DC0 2D         2370          DEC  L
9DC1 DD6604     2380          LD   H,(IX+4)
9DC4 25         2390          DEC  H
9DC5 CD1ABC     2400          CALL CHRPOS
9DC8 DD4600     2410          LD   B,(IX+0)
9DCB CB20       2420          SLA  B
9DCD 05         2430          DEC  B
9DCE 0E08       2440          LD   C,8
9DD0 E5         2450 ROWL     PUSH HL
9DD1 C5         2460          PUSH BC
9DD2 CB26       2470          SLA  (HL)
9DD4 7E         2480          LD   A,(HL)
9DD5 F5         2490          PUSH AF
9DD6 5D         2500 BYTEL    LD   E,L
9DD7 54         2510          LD   D,H
9DD8 2C         2520          INC  L
9DD9 200A       2530          JR   NZ,HLLOK
9DDB 24         2540          INC  H
9DDC 7C         2550          LD   A,H
9DDD E607       2560          AND  7
9DDF 2004       2570          JR   NZ,HLLOK
9DE1 7C         2580          LD   A,H
9DE2 D608       2590          SUB  8
9DE4 67         2600          LD   H,A
9DE5 1A         2610 HLLOK    LD   A,(DE)
9DE6 CB26       2620          SLA  (HL)
9DE8 3804       2630          JR   C,SETBTL
9DEA CBA7       2640          RES  4,A
9DEC 1802       2650          JR   TSTBTL
9DEE CBE7       2660 SETBTL   SET  4,A
9DF0 CB66       2670 TSTBTL   BIT  4,(HL)
9DF2 2802       2680          JR   Z,BITLOK
9DF4 CBC7       2690          SET  0,A
9DF6 12         2700 BITLOK   LD   (DE),A
9DF7 10DD       2710          DJNZ BYTEL
9DF9 F1         2720          POP  AF
```

```
9DFA 3804      2730          JR   C,WRAPL
9DFC CBA6      2740          RES  4,(HL)
9DFE 1802      2750          JR   ENDPXL
9E00 CBE6      2760 WRAPL    SET  4,(HL)
9E02 CB67      2770 ENDPXL   BIT  4,A
9E04 2802      2780          JR   Z,ROWFNL
9E06 CBC6      2790          SET  0,(HL)
9E08 C1        2800 ROWFNL   POP  BC
9E09 E1        2810          POP  HL
9E0A 7C        2820          LD   A,H
9E0B C608      2830          ADD  A,8
9E0D 67        2840          LD   H,A
9E0E 0D        2850          DEC  C
9E0F 20BF      2860          JR   NZ,ROWL
9E11 C9        2870          RET
9E12 FE03      2880 RWRAP    CP   3
9E14 C0        2890          RET  NZ
9E15 DD6E02    2900          LD   L,(IX+2)
9E18 2D        2910          DEC  L
9E19 DD4E00    2920          LD   C,(IX+0)
9E1C DD7E04    2930          LD   A,(IX+4)
9E1F 81        2940          ADD  C
9E20 D602      2950          SUB  2
9E22 67        2960          LD   H,A
9E23 CD1ABC    2970          CALL CHRPOS
9E26 23        2980          INC  HL
9E27 41        2990          LD   B,C
9E28 CB20      3000          SLA  B
9E2A 05        3010          DEC  B
9E2B 0E08      3020          LD   C,8
9E2D E5        3030 ROWR     PUSH HL
9E2E C5        3040          PUSH BC
9E2F CB3E      3050          SRL  (HL)
9E31 7E        3060          LD   A,(HL)
9E32 F5        3070          PUSH AF
9E33 5D        3080 BYTER    LD   E,L
```

```
9E34 54         3090        LD   D,H
9E35 7D         3100        LD   A,L
9E36 2D         3110        DEC  L
9E37 B7         3120        OR   A
9E38 200A       3130        JR   NZ,HLROK
9E3A 7C         3140        LD   A,H
9E3B 25         3150        DEC  H
9E3C E607       3160        AND  7
9E3E 2004       3170        JR   NZ,HLROK
9E40 7C         3180        LD   A,H
9E41 C608       3190        ADD  A,8
9E43 67         3200        LD   H,A
9E44 1A         3210 HLROK  LD   A,(DE)
9E45 CB3E       3220        SRL  (HL)
9E47 3804       3230        JR   C,SETBTR
9E49 CB9F       3240        RES  3,A
9E4B 1802       3250        JR   TSTBTR
9E4D CBDF       3260 SETBTR SET  3,A
9E4F CB5E       3270 TSTBTR BIT  3,(HL)
9E51 2802       3280        JR   Z,BITROK
9E53 CBFF       3290        SET  7,A
9E55 12         3300 BITROK LD   (DE),A
9E56 10DB       3310        DJNZ BYTER
9E58 F1         3320        POP  AF
9E59 3804       3330        JR   C,WRAPR
9E5B CB9E       3340        RES  3,(HL)
9E5D 1802       3350        JR   ENDPXR
9E5F CBDE       3360 WRAPR  SET  3,(HL)
9E61 CB5F       3370 ENDPXR BIT  3,A
9E63 2802       3380        JR   Z,ROWFNR
9E65 CBFE       3390        SET  7,(HL)
9E67 C1         3400 ROWFNR POP  BC
9E68 E1         3410        POP  HL
9E69 7C         3420        LD   A,H
9E6A C608       3430        ADD  A,8
9E6C 67         3440        LD   H,A
```

```
9E6D 0D       3450        DEC  C
9E6E 20BD     3460        JR   NZ,ROWR
9E70 C9       3470        RET
9E71 DD4E00   3480 GRPNT4 LD   C,(IX+0)
9E74 DD6E02   3490        LD   L,(IX+2)
9E77 DD6603   3500        LD   H,(IX+3)
9E7A DD5E04   3510        LD   E,(IX+4)
9E7D DD5605   3520        LD   D,(IX+5)
9E80 0602     3530        LD   B,2
9E82 D5       3540 NXTCDE PUSH DE
9E83 E5       3550        PUSH HL
9E84 C5       3560        PUSH BC
9E85 CDC0BB   3570        CALL MOVABS
9E88 C1       3580        POP  BC
9E89 79       3590        LD   A,C
9E8A 0C       3600        INC  C
9E8B C5       3610        PUSH BC
9E8C CDFCBB   3620        CALL GRACHR
9E8F C1       3630        POP  BC
9E90 79       3640        LD   A,C
9E91 0C       3650        INC  C
9E92 C5       3660        PUSH BC
9E93 CDFCBB   3670        CALL GRACHR
9E96 C1       3680        POP  BC
9E97 E1       3690        POP  HL
9E98 D1       3700        POP  DE
9E99 3E10     3710        LD   A,16
9E9B 2B       3720 MOVEGR DEC  HL
9E9C 3D       3730        DEC  A
9E9D 20FC     3740        JR   NZ,MOVEGR
9E9F 10E1     3750        DJNZ NXTCDE
9EA1 C9       3760        RET
              3770        END
```

The OUTs and PEEKs and POKEs and CALLs of the Amstrad

The mammoth title says it all. This chapter is devoted to a number of useful routines and features that can be accessed via the above commands. We have so far discussed all of these commands and their uses but have only given a small number of examples. Here are the rest.

1) There are a number of ways to prevent other people looking at your listings but most rely on the user not being able to break into the program. This POKE allows the program to be broken but will not let the actual listing process take place. This is because the first program line is POKEd with a control character.
   Make Line 1: 1 REM***
   Then type POKE &176, 255.
   Your program should now be unlistable.
2) A quick CALL here, to flush the keyboard buffer type: CALL &BFF9, &C9, &1CED, &CF00
3) I don't know how many of you have found it annoying having to set up a LOAD or CAT every time you wish to position the cassette's tape using the play button. OUT 512,6 enables the cassette play button, allowing you to use it again, while OUT 512,0 disables it again.
4) A simple form of horizontal scrolling can be obtained with the single command OUT 256,N, where N is the character position. Thus to scroll from the middle of the screen in mode 1, try OUT 256,20. The scroll produces a 'wrap-around' effect where the characters sliding off the left side of the screen reappear on the right.
5) To disable the cursor on the screen, type CALL &BB7E. To enable the cursor, type CALL &BB18
6) CALL &BB18 is equal to the BASIC commands:
   WHILE INKEY$ = "": WEND
   This program line and the call both wait for a key to be pressed.
7) CALL &BD19 performs a task known as frame flyback. This simply gives your animated screen displays a better looking, smoother movement.

Assemblers and Assembly Language

You will have noticed that not only are BASIC listings included in this book but also listings in Assembly language. For those that are unfamiliar with this language it is simply a method of explaining the various Machine Code instructions that are available. For example, one Machine Code instruction makes the computer return to BASIC from Machine Code. In the BASIC loader programs this is shown as C9, as the BASIC loaders use the hexadecimal (base 16) numbering system. The equivalent Assembly language code is RET, which is obviously short for RETURN. Have a look at one of the Assembly language listings and you will see that RET is the penultimate instruction in the listing.

For more details of Assembly language, I suggest you have a look at one of the guides mentioned in the bibliography. What I am interested in mentioning here is a further set of instructions. This further set can be seen in the Assembly language listings. Such instructions as END, ORG, DEFM and PRNT are all commands used by the assembler (the program that allows you to write Machine Code in Assembly language). Using an assembler makes Machine Code easier to write and easier to follow another person's routine.

These assembler commands, or directives as they are known, aid the Machine Code programmer and have to be included within the Assembly language listing. So as not to confuse you when looking at the Assembly language listings, they are all detailed below:

ORG END EQU DEFL DEFB DEFW
DEFW DEFS DEFM PRNT ENT

Bibliography

There are a number of books on Z80 and Amstrad-specific Z80 Machine Code programming. We have chosen several titles to look at.

Amstrad Machine Language For The Absolute Beginner
by Joe Pritchard. Melbourne House. ISBN 0 86161 193 4

In our opinion, the best of the tutorial guides on the subject (before ours was published of course!). Seriously though, the book is a very competent guide for the beginner to Machine Code on the Amstrad. Unlike a number of Machine Code guides it is very Amstrad-specific and includes a number of the available ROM calls. The presentation of the book is poor; just printer outputs and the odd diagram. There are a number of spelling mistakes in the text, but more worryingly there are also a few errors in the program listings. But all in all, a good buy for those wishing to go further than first principles with Machine Code.

Amstrad CPC 464 Machine Code
by Steve Webb. Virgin Books. ISBN 0 86369 082 3

This slim book is ideal for the complete beginner to Machine Code who thinks that PUSHing and POPing are new forms of dancing! Instead of launching into complex explanations of Machine Code concepts and principles, the book stays on a more practical level and introduces you to some of the Z80 Op codes that you may use. You would need to buy another additional book to supplement the teachings of this guide but for the absolute beginner it is well worth considering.

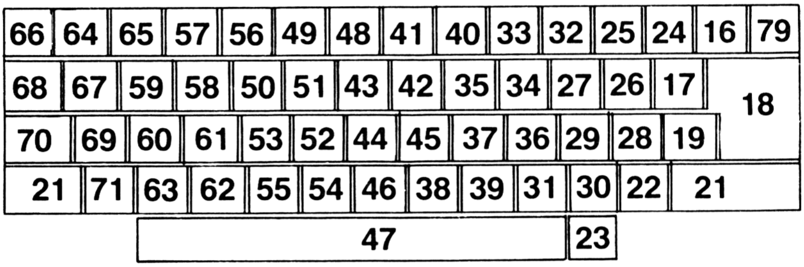
The Concise Firmware Specification
Order Code: SOFT 158. Amsoft.

At £20 for a loose-leaf folder of information, you may be mistaken in thinking that Amsoft are ripping the public off. This is not the case. For your money, you get a complete technical reference guide which gives details of all the ROM routines, how they work and what parameters they require. A number of the routines in our book would have been far more difficult to write if we had not had this guide.

Keyboard Layout

The following diagram shows the numerical code of all the keys:

MAIN KEYBOARD



NUMERIC KEYPAD

10	11	3
20	12	4
13	14	5
15	7	6

CURSOR KEYS

	0	
8	9	1
	2	

MACHINE CODE *Routines* for Your Amstrad

BASIC is the simplest of computer languages to learn, and it is usually quickly mastered by the newcomer to computing. As they progress, many programmers find that BASIC simply cannot cope with the parts of their programs which require speedy operation, such as fast screen movement.

Now you can harness the true power of your Amstrad through Machine Code which can speed up your programs, with impressive results.

This book contains over 20 routines, each written so that you can incorporate them within your own programs with the minimum of fuss and bother. There is no need for an Assembler or any knowledge of Machine Code.

Each routine is fully explained and comes with examples and suggestions for its use. With this book, you will be able to achieve stunning effects, such as fast and smooth screen-scrolling; multicoloured and rotating characters; new sound effects commands; italic printing and more – routines which are impossible in BASIC, and which will enhance your tired programs, be they for games, educational, or business applications.

Virgin

ISBN 0 86369 131 5
£4.99

00499
9 780863 691317

Machine Code Routines for your Amstrad Gifford and Vincent

AMSTRAD CPC

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA

https://acpc.me/